Docker 容器启动 Dubbo 服务

作者	日期
姜鹏	2018/09/06

微信号：comeon_betty

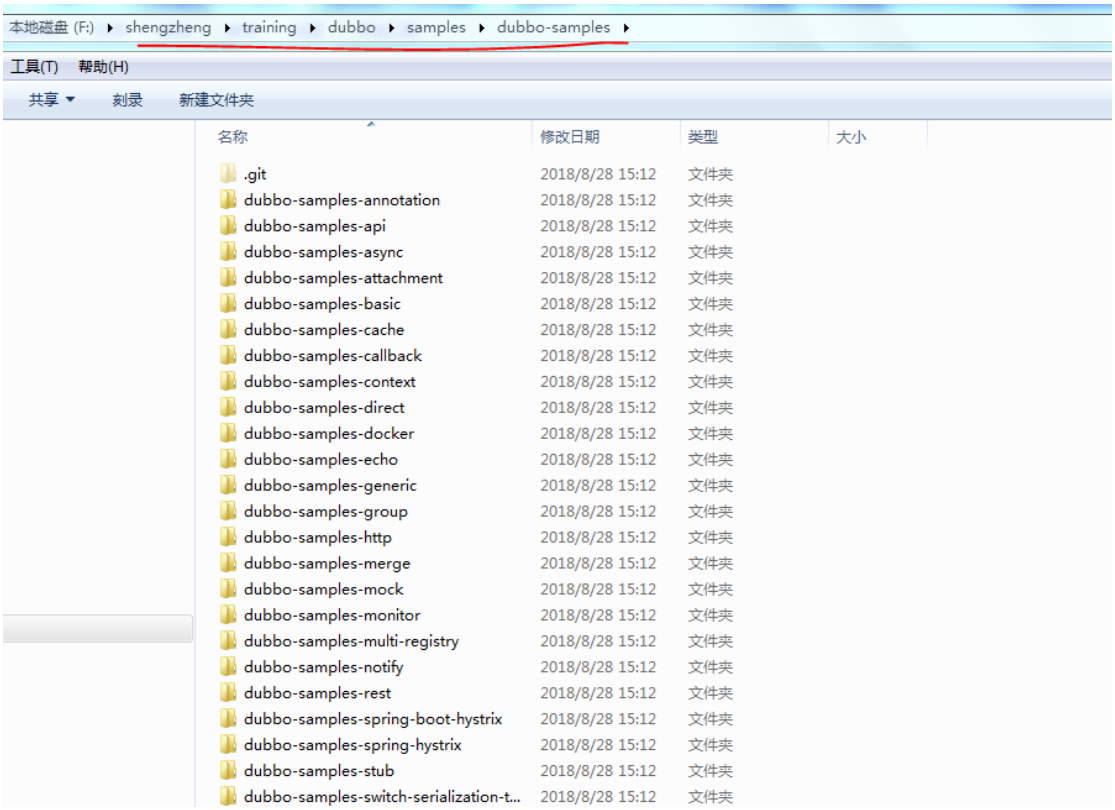
云主机操作系统：

```
[root@izwz97ihkr7f1efzto963sz ~]#  cat /etc/redhat-release
CentOS Linux release 7.4.1708 (Core)
```

1 dubbo-samples-rest

https://github.com/dubbo/dubbo-samples/tree/master/dubbo-samples-rest

$git clone https://github.com/dubbo/dubbo-samples



$export MAVEN_HOME=F:/software/apache-maven-3.5.2

$MAVEN_HOME/bin/mvn package -X -DskipTests

2 zookeeper Docker 安装

Docker 中 搜索 zookeeper

$docker search zookeeper

```
[root@izwz97ihkr7flefzto963sz ~]# docker search zookeeper
INDEX       NAME                                      DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
docker.io   docker.io/zookeeper                       Apache ZooKeeper is an open-source server ...   447       [OK]
docker.io   docker.io/jplock/zookeeper                Builds a docker image for Zookeeper versio...   161                  [OK]
docker.io   docker.io/mesoscloud/zookeeper            ZooKeeper                                       73                   [OK]
docker.io   docker.io/wurstmeister/zookeeper                                                          57                   [OK]
docker.io   docker.io/confluentinc/cp-zookeeper       Official Confluent Docker Image for Zookeeper   30
docker.io   docker.io/mbabineau/zookeeper-exhibitor                                                   23                   [OK]
docker.io   docker.io/digitalwonderland/zookeeper     Latest Zookeeper - clusterable                  18                   [OK]
docker.io   docker.io/confluent/zookeeper                                                             12                   [OK]
docker.io   docker.io/tobilg/zookeeper-webui          Docker image for using `zk-web` as ZooKeep...   10                   [OK]
docker.io   docker.io/debezium/zookeeper              Zookeeper image required when running the ...   8                    [OK]
docker.io   docker.io/hyperledger/fabric-zookeeper    Fabric Zookeeper docker image for Hyperled...   8
docker.io   docker.io/elevy/zookeeper                 ZooKeeper configured to execute an ensembl...   6                    [OK]
docker.io   docker.io/springxd/zookeeper              A Docker image that can run a ZooKeeper se...   6                    [OK]
docker.io   docker.io/31z4/zookeeper                  Dockerized Apache Zookeeper.                    5                    [OK]
docker.io   docker.io/thefactory/zookeeper-exhibitor  Exhibitor-managed ZooKeeper with S3 backup...   5                    [OK]
docker.io   docker.io/bitnami/zookeeper               ZooKeeper is a centralized service for dis...   4                    [OK]
docker.io   docker.io/ciscocloud/zookeeper                                                            1                    [OK]
docker.io   docker.io/openshift/zookeeper-346-fedora20  ZooKeeper 3.4.6 with replication support     1
docker.io   docker.io/paulbrown/zookeeper             Zookeeper on Kubernetes (PetSet)                1                    [OK]
docker.io   docker.io/avvo/zookeeper                  Apache Zookeeper                                0                    [OK]
docker.io   docker.io/duffqiu/zookeeper-cli                                                           0                    [OK]
docker.io   docker.io/jamiecressey89/marathon-zookeeper  Zookeeper image that uses Marathon's API f...  0                    [OK]
docker.io   docker.io/midonet/zookeeper               Dockerfile for a Zookeeper server.              0                    [OK]
docker.io   docker.io/phenompeople/zookeeper          Apache ZooKeeper is an open-source server ...   0                    [OK]
docker.io   docker.io/strimzi/zookeeper                                                               0
```

$docker pull zookeeper

将下载的文件保存在本地镜像：

$docker save -o /usr/home/softwares/images/zookeeper-image.tar

docker.io/zookeeper

```
-rw------- 1 root root 153476608 Aug 28 15:31 zookeeper-image.tar
[root@izwz97ihkr7flefzto963sz images]# pwd
/usr/home/softwares/images
```

将 zookeeper-image.tar 载入镜像

docker load -i /usr/home/softwares/images/zookeeper-image.tar

启动 zookeeper

$docker run --name zookeeper-1 --restart always -d
docker.io/zookeeper

查看正在运行的容器：

$docker ps

```
[root@izwz97ihkr7flefzto963sz images]# docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS                            NAMES
9c16896bdcfe        docker.io/zookeeper    "/docker-entrypoin..."   About a minute ago  Up About a minute   2181/tcp, 2888/tcp, 3888/tcp     zookeeper-1
```

启动 zookeeper client

$ docker exec -it 9c16896bdcfe zkCli.sh

```
[root@izwz97ihkr7flefzto963sz images]# docker exec -it 9c16896bdcfe zkCli.sh
Connecting to localhost:2181
2018-08-28 07:42:40,132 [myid:] - INFO  [main:Environment@100] - Client environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2
18 04:05 GMT
2018-08-28 07:42:40,136 [myid:] - INFO  [main:Environment@100] - Client environment:host.name=9c16896bdcfe
2018-08-28 07:42:40,137 [myid:] - INFO  [main:Environment@100] - Client environment:java.version=1.8.0_171
2018-08-28 07:42:40,140 [myid:] - INFO  [main:Environment@100] - Client environment:java.vendor=Oracle Corporation
2018-08-28 07:42:40,140 [myid:] - INFO  [main:Environment@100] - Client environment:java.home=/usr/lib/jvm/java-1.8-openjdk/jre
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:java.class.path=/zookeeper-3.4.13/bin/../build/classes:/zookeeper-3.4.13/bin/../bu
d/lib/*.jar:/zookeeper-3.4.13/bin/../lib/slf4j-log4j12-1.7.25.jar:/zookeeper-3.4.13/bin/../lib/slf4j-api-1.7.25.jar:/zookeeper-3.4.13/bin/../lib/netty-3.10.6.Final.ja
/zookeeper-3.4.13/bin/../lib/log4j-1.2.17.jar:/zookeeper-3.4.13/bin/../lib/jline-0.9.94.jar:/zookeeper-3.4.13/bin/../lib/audience-annotations-0.5.0.jar:/zookeeper-3.4.
3/bin/../zookeeper-3.4.13.jar:/zookeeper-3.4.13/bin/../src/java/lib/*.jar:/conf:
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:java.library.path=/usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/server:/usr/lib/jvm/j
va-1.8-openjdk/jre/lib/amd64:/usr/lib/jvm/java-1.8-openjdk/jre/../lib/amd64:/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:java.io.tmpdir=/tmp
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:java.compiler=<NA>
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:os.name=Linux
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:os.arch=amd64
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:os.version=3.10.0-693.2.2.el7.x86_64
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:user.name=root
2018-08-28 07:42:40,141 [myid:] - INFO  [main:Environment@100] - Client environment:user.home=/root
2018-08-28 07:42:40,142 [myid:] - INFO  [main:Environment@100] - Client environment:user.dir=/zookeeper-3.4.13
2018-08-28 07:42:40,143 [myid:] - INFO  [main:ZooKeeper@442] - Initiating client connection, connectString=localhost:2181 sessionTimeout=30000 watcher=org.apache.zooke
per.ZooKeeperMain$MyWatcher@4b85612c
Welcome to ZooKeeper!
2018-08-28 07:42:40,170 [myid:] - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@1029] - Opening socket connection to server localhost/127.0.0.1:2181. W
l not attempt to authenticate using SASL (unknown error)
JLine support is enabled
2018-08-28 07:42:40,275 [myid:] - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@879] - Socket connection established to localhost/127.0.0.1:2181, initia
ing session
[zk: localhost:2181(CONNECTING) 0] 2018-08-28 07:42:40,317 [myid:] - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@1303] - Session establishment complet
 on server localhost/127.0.0.1:2181, sessionid = 0x100632113950000, negotiated timeout = 30000

WATCHER::
```

找到安装路径：

$docker exec -it 9c16896bdcfe /bin/bash

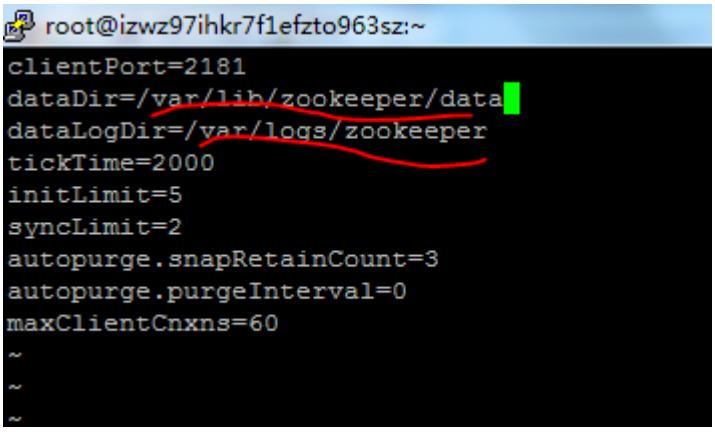
$pwd

停止服务：

$/zookeeper-3.4.13/bin/zkCli.sh stop

```
bash-4.4# ./zkServer.sh stop
ZooKeeper JMX enabled by default
Using config: /conf/zoo.cfg
Stopping zookeeper ... no zookeeper to stop (could not find file /data/zookeeper_server.pid)
```

需要配置：/conf/zoo.cfg



$mkdir -p /var/lib/zookeeper/data

$mkdir -p /var/logs/zookeeper

$touch /var/lib/zookeeper/data/zookeeper_server.pid

启动服务端：

$./zkCli.sh -server 127.0.0.1:2181

如果集群，配置文件修改如下：

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/home/dubbo/zookeeper-3.3.3/data

clientPort=2181

server.1=10.20.153.10:2555:3555

server.2=10.20.153.11:2555:3555

使用：

dubbo.registry.address=zookeeper://10.20.153.10:2181?backup=10.20.153.11:2181

或者：

<dubbo:registry protocol="zookeeper"
address="10.20.153.10:2181,10.20.153.11:2181" />

3 zookeeper 官方教程

http://zookeeper.apache.org/doc/current/index.html

3.1 什么是 zookeeper?

分布应用的分布式协调服务。

3.2 zookeeper 特点

- 支持副本集：

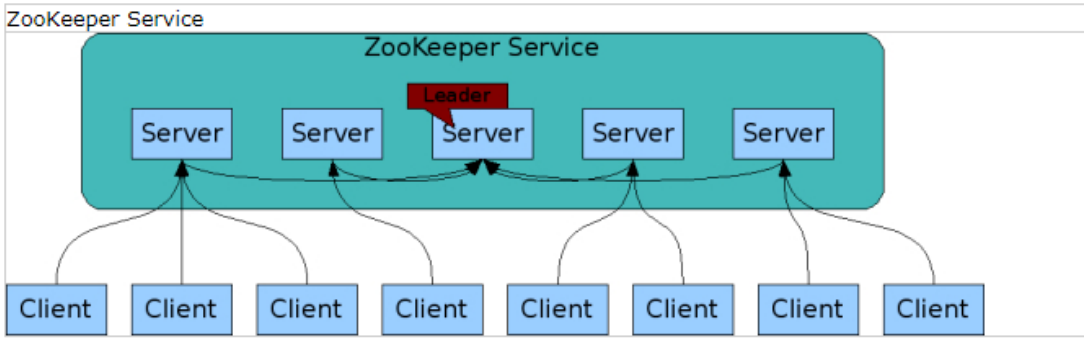


图 3.1 zookeeper 副本集

- 组成 ZooKeeper 服务的服务器必须相互之间能够识别。

维护一个内存的镜像状态，持久存储事务日志和快照。只要主节点在，ZooKeeper 服务就可用。

- 客户端连接到单个 ZooKeeper 服务，维持 TCP 连接，发送请求，获取响应和观测事件、发送心跳。如果 TCP 连接终端。客户端将连接到不同的服务节点。

- ZooKeeper 有次序

  ZooKeeper 通过一个数字标记彼此的更新，这个数字反映了所有 ZooKeeper 的事务。顺次的操作可以使用顺序去实现高阶的抽象，比如：同步化的原型。

- 快速

  ZooKeeper 在“读领域”负载尤其的快，ZooKeeper 应用运行在成千上万的机器上，在读比写更普遍的场合性能发挥到极致， 如：读写比例 10:1 左右。

- Data 模型和分层命名空间

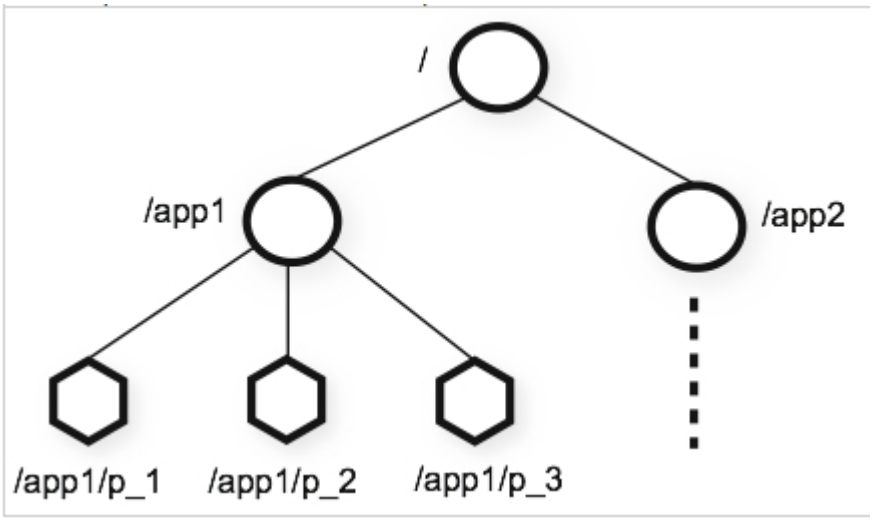


图 3.2 zookeeper 分层命名空间

- 节点和短暂节点（ephemeral nodes）

这儿以 term Znodes 来澄清节点的概念。

Znodes 维持一个 stat 结构， 该结构包含数据变化的版本，ACL（访问控制列表）变化和时间戳， 以允许缓存校验和协调更新。每当 znode 数据变化，版本数也增加。例如：任何时候客户端取回的数据的同时也取回了数据的版本。

在命名空间存储在每个 znode 自动被读写，读获取所有与 znode 相关的数据字节而写则取代所有的数据。

ZooKeeper 也有瞬时节点（ephemeral nodes）的概念，只要创建 znode 的会话是活跃的，这些瞬时节点都将存在。当会话终止，瞬时节点将删除。

- 条件更新和观察

客户端可以在 znodes 上设置观察。当客户端收到一个数据包说“znode 已经变化了”，观察将触发。当客户端和任意 Zoo Keeper 服务断开，客户端将收到位置提醒。

- Zookeeper 的保障

Sequential Consistency : Updates from a client will be applied in the order that they were sent.

Atomicity : Updates either succeed or fail. No partial results.

Single System Image : A client will see the same view of the service regardless of the server that it connects to.

Reliability : Once an update has been applied, it will persist from that time forward until a client overwrites the update.

Timeliness - The clients view of the system is guaranteed to be up-to-date within a certain time bound.

3.2 Zookeeper Standardalone 操作

http://zookeeper.apache.org/doc/current/zookeeperStarted.html#sc_InstallingSingleMode

3.2.1 /zookeeper-3.4.13/conf/zoo.cfg

```
tickTime=2000
dataDir=/var/lib/zookeeper
clientPort=2181
```

3.2.2 非 Docker 容器启动

```
$ bin/zkCli.sh -server 127.0.0.1:2181
```

https://blog.csdn.net/yuanlaijike/article/details/79654183#%E4%BA%8C%E6%90%AD%E5%BB%BAzookeeper

3.2.3 windows zookeeper

下载：zookeeper-3.4.10.tar.gz，解压：

启动服务端：

set JAVA_HOME=F:/software/JDK

set ZOOKEEPER_HOME=F:/software/zookeeper-3.4.10/zookeeper-3.4.10

%ZOOKEEPER_HOME%/bin/zkServer start

4 安装 dubbo-admin.war

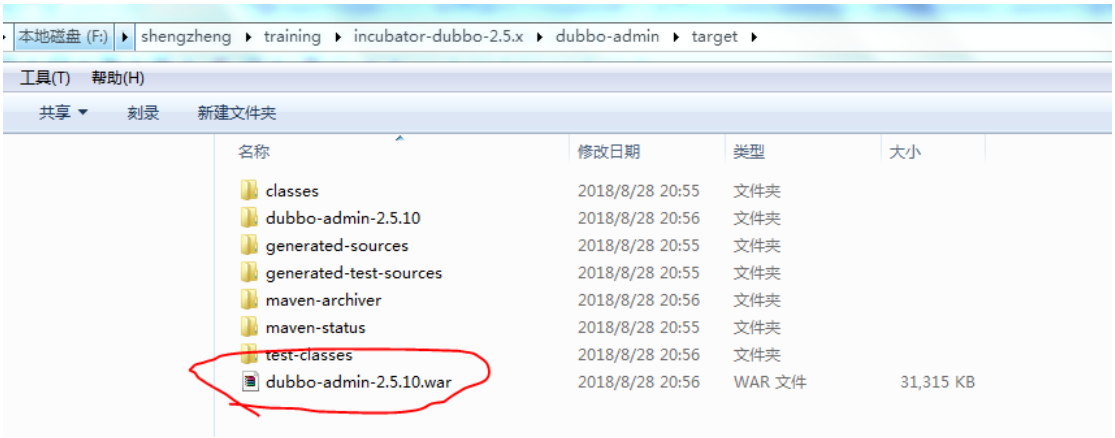
4.1 download source

https://github.com/apache/incubator-dubbo/releases

4.2 maven compile

```
$export MAVEN_HOME=F:/software/apache-maven-3.5.2
$export JAVA_HOME=F:/software/JDK
$MAVEN_HOME/bin/mvn package -DskipTests -X
```

编译完后：产物 dubbo-admin-2.5.10.war



4.2 Docker 启动 Tomcat

```
$docker run --name tomcat-1 --restart always -d -p 8081:8080
docker.io/tomcat
```

4.3 upload to cloud

上传文件到云，比如：

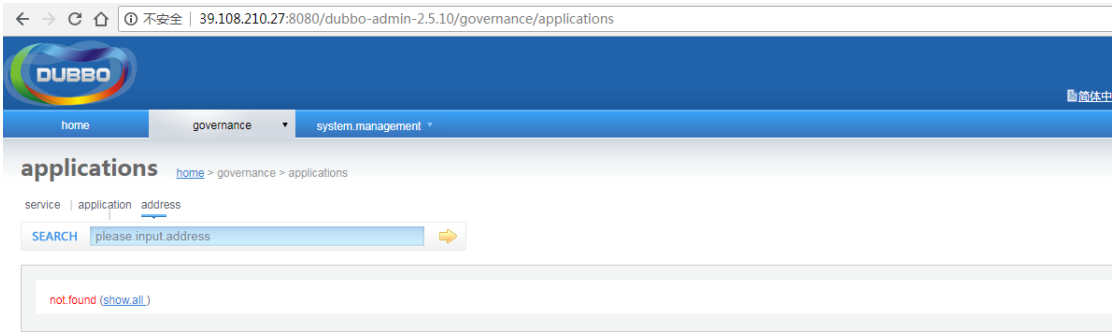
```
$docker cp /root/dubbo-admin-2.5.10.war  \
tomcat-1:/usr/local/tomcat/webapps/
```

```
$docker run --name tomcat-1 --restart always -d -p 8081:8080
docker.io/tomcat
```

- 以挂载的形式启动 Tomcat

$docker run -d -v /root/dubbo-admin-2.5.10.war:/usr/local/tomcat/webapps/dubbo-admin-2.5.10.war -p 8080:8080 docker.io/tomcat

登陆后台：root/root



如果通过

$netstat -atnp|grep 8080 杀掉 Tomcat，发现很快又重启了。

5 API 调试

https://www.jianshu.com/p/70151fc0ef5d

6 .YAML 标记语言

http://www.ruanyifeng.com/blog/2016/07/yaml.html?f=tt

7 Dockerfile 创建镜像

https://www.cnblogs.com/jie-fang/p/7927643.html

指令　说明

FROM　指定所创建镜像的基础镜像

MAINTAINER　指定维护者信息

RUN　运行命令

CMD　指定启动容器时默认执行的命令

LABEL 指定生成镜像的元数据标签信息

EXPOSE　声明镜像内服务所监听的端口

ENV　指定环境变量

ADD　赋值指定的<src>路径下的内容到容器中的<dest>路径下，<src>可以为 URL；如果为 tar 文件，会自动解压到<dest>路径下

COPY　赋值本地主机的<scr>路径下的内容到容器中的<dest>路径下；一般情况下推荐使用 COPY 而不是 ADD

ENTRYPOINT　指定镜像的默认入口

VOLUME　创建数据挂载点

USER　指定运行容器时的用户名或 UID

WORKDIR　配置工作目录

ARG　指定镜像内使用的参数(例如版本号信息等)

ONBUILD　配置当前所创建的镜像作为其他镜像的基础镜像时，所执行的创建操作的命令

STOPSIGNAL　容器退出的信号

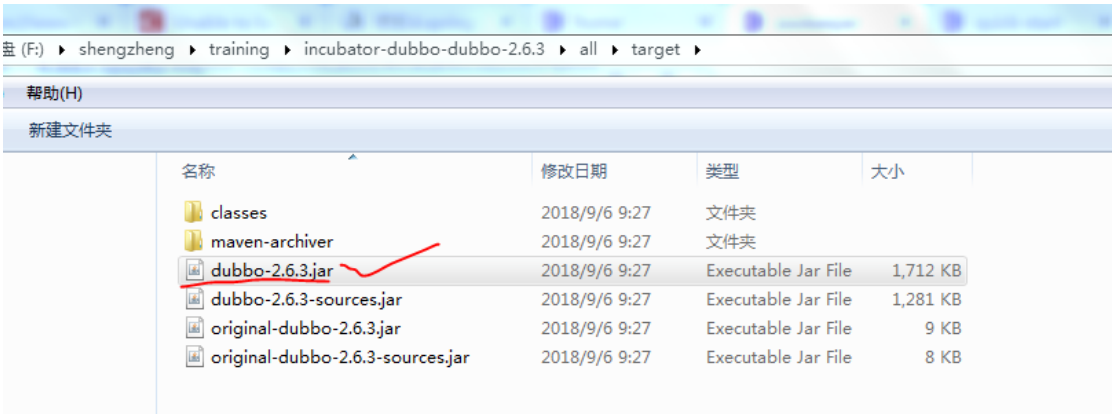
HEALTHCHECK 如何进行健康检查

SHELL 指定使用 SHELL 时的默认 SHELL 类型

8 基于源码安装 Dubbo

```
cd F:\shengzheng\training\incubator-dubbo-dubbo-2.6.3

set JAVA_HOME=F:\software\JDK

set MAVEN_HOME=F:\software\apache-maven-3.5.2

set MAVEN_OPTS=-Xmx1024m -XX:MaxPermSize=512m

%MAVEN_HOME%/bin/mvn install -DskipTests -X
```

成功安装：



9 践行官方实例

9.1 使用多路广播中心发现服务。

不使用 zookeeper 完成服务注册，使用多路广播中心。

9.2 工程依赖

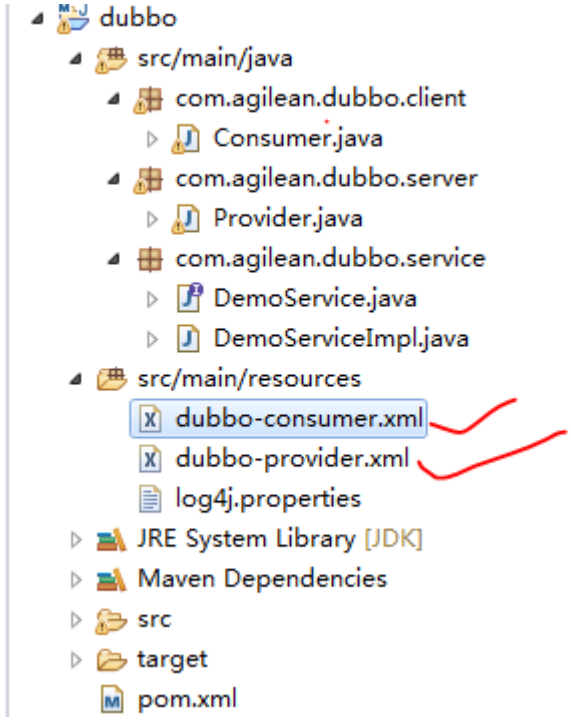
工程中引入：dubbo 包，章节 8 中基于源码安装的 jar。必须去除 dubbo 自带的老的 spring 依赖。

```
 <dependency>
<groupId>com.alibaba</groupId>
<artifactId>dubbo</artifactId>
<version>2.6.3</version>
<exclusions>
            <exclusion>
                <artifactId>spring</artifactId>
                <groupId>org.springframework</groupId>
            </exclusion>
  </exclusions>

  </dependency>
```

导入:spring4 依赖。使用 4.0.8.RELEASE 版本。

```
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${org.springframework.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
        <version>${org.springframework.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${org.springframework.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${org.springframework.version}</version>
</dependency>
```

9.3 工程结构



9.3.1 服务提供者配置及启动

```
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
       http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">

    <!-- provider's application name, used for tracing dependency relationship -->
    <dubbo:application name="demo-provider"/>
    <!-- use multicast registry center to export service -->
    <!--   <dubbo:registry address="multicast://224.5.6.7:1234"/> -->
    <dubbo:registry address="multicast://224.5.6.7:1234"/>
    <!-- use dubbo protocol to export service on port 20880 -->
    <dubbo:protocol name="dubbo" port="20880"/>
    <!-- service implementation, as same as regular local bean -->
    <bean id="demoService" class="com.agilean.dubbo.service.DemoServiceImpl"/>
    <!-- declare the service interface to be exported -->
    <dubbo:service interface="com.agilean.dubbo.service.DemoService" ref="demoService"/>
</beans>
```

```java
package com.agilean.dubbo.server;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Provider {
  public static void main(String[] args) throws Exception{

      System.setProperty("java.net.preferIPv4Stack", "true");
      ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext(new String[]
              {"dubbo-provider.xml"});
      context.start();
      System.out.println("Provider started.");
      System.in.read(); // press any key to exit
  }
}
```

9.3.1 消费者配置及调用服务

```xml
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/sc
       http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">

    <!-- consumer's application name, used for tracing dependency relationship (not a matching criterior
    don't set it same as provider -->
    <dubbo:application name="demo-consumer"/>
    <!-- use multicast registry center to discover service -->
    <dubbo:registry address="multicast://224.5.6.7:1234"/>
    <!-- generate proxy for the remote service, then demoService can be used in the same way as the
    local regular interface -->
    <dubbo:reference id="demoService" check="false" interface="com.agilean.dubbo.service.DemoService"/>
</beans>
```

```java
package com.agilean.dubbo.client;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Consumer {

    public static void main(String[] args) {
            ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext(new String[]
                    {"dubbo-consumer.xml"});
            context.start();
            // Obtaining a remote service proxy
            DemoService demoService = (DemoService)context.getBean("demoService");
            // Executing remote methods
            String hello = demoService.sayHello("world");
            // Display the call result
            System.out.println(hello);

    }

}
```