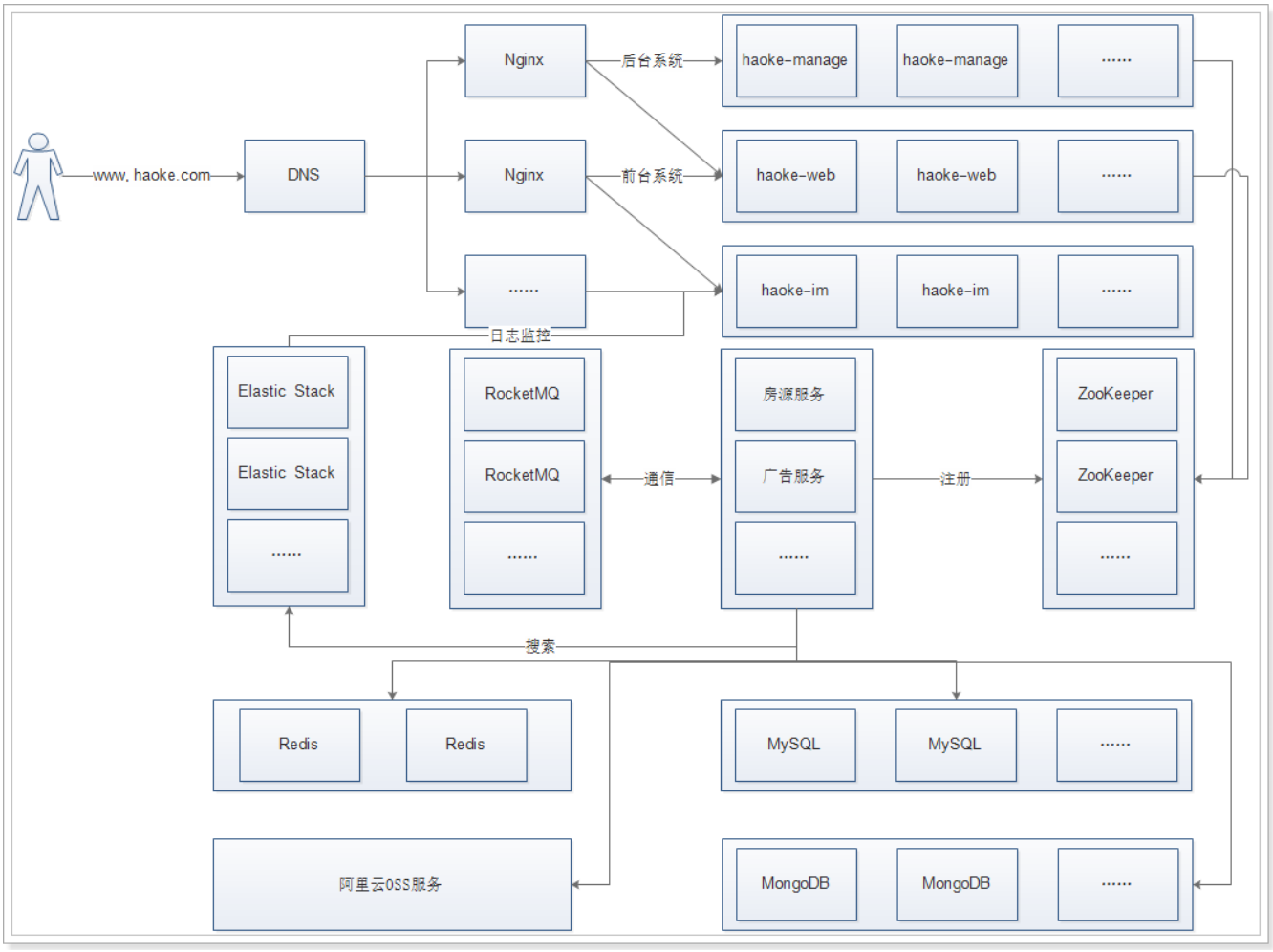
课程说明

- 部署架构
- 部署计划
- 实施部署
- 打包项目
- 功能测试

1、部署架构



说明：

- 在架构中集群的节点数根据实际情况设置
- 项目的中的实际系统并没有完全展示出来

2、部署计划

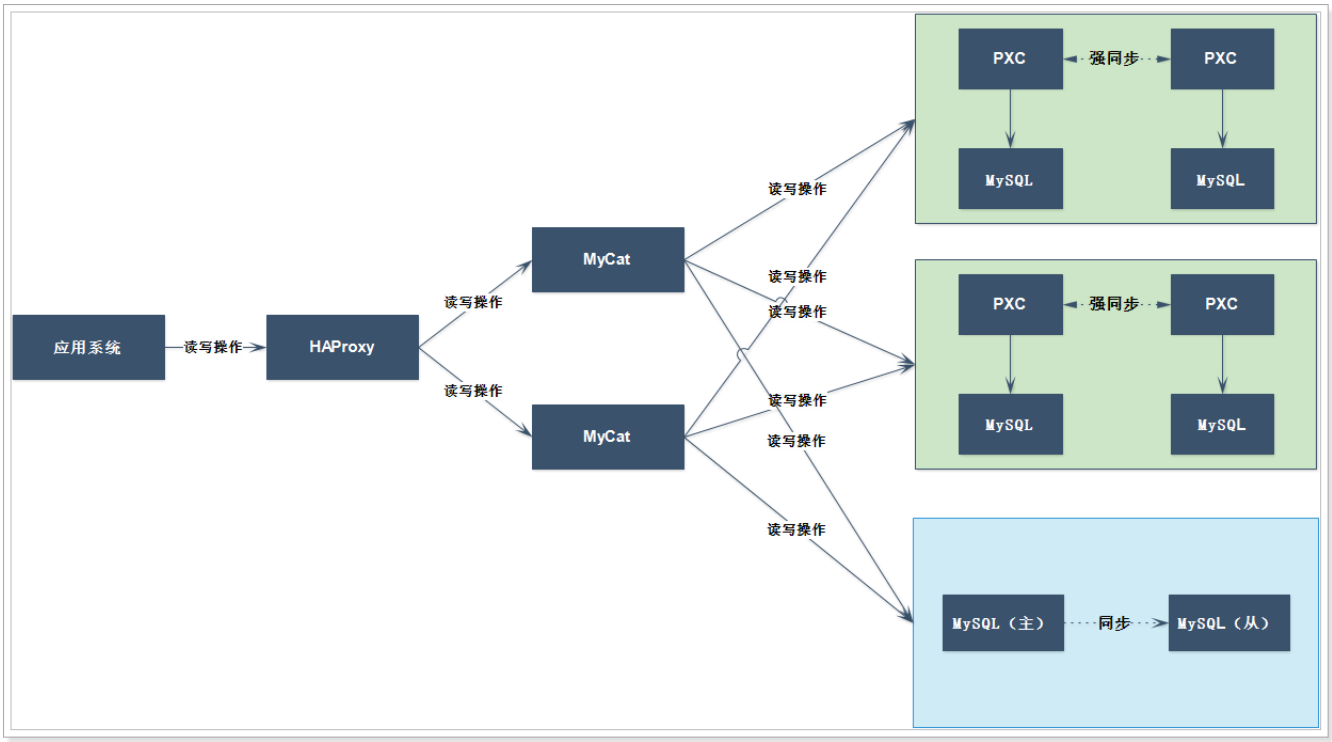
在实际项目中，在部署上线之前需要对所有的服务进行盘点，然后根据用户数以及并发数，对需要的服务器进行统计，然后进行采购服务器，最后实施部署。

由于我们处于学习阶段，服务器资源有限，所以需要在现有的服务器资源上进行分配。

服务器资源目前拥有3台服务器，分别是192.168.1.7、192.168.1.18、192.168.1.19。

2.1、MySQL服务

2.1.1、架构



2.1.2、规划

服务	端口	服务器	容器名
MySQL-node01	13306	192.168.1.18	pxc_node1
MySQL-node02	13307	192.168.1.18	pxc_node2
MySQL-node03	13308	192.168.1.18	pxc_node3
MySQL-node04	13309	192.168.1.18	pxc_node4
MySQL-node05	13310	192.168.1.19	ms_node1
MySQL-node06	13311	192.168.1.19	ms_node2
MyCat-node01	11986，18068，19068	192.168.1.19	mycat_node01
MyCat-node02	11987，18069，19069	192.168.1.19	mycat_node02
HAProxy	4001，4002	192.168.1.19	haproxy

2.1.3、实施

2.1.3.1、部署pxc集群

```
#创建数据卷（存储路径：/var/lib/docker/volumes）
docker volume create haoke-v1
docker volume create haoke-v2
docker volume create haoke-v3
docker volume create haoke-v4
docker volume create haoke-v5
docker volume create haoke-v6

#拉取镜像
docker pull percona/percona-xtradb-cluster:5.7
docker tag percona/percona-xtradb-cluster:5.7 pxc


#创建网络
docker network create --subnet=172.30.0.0/24 pxc-network
```

```
#集群1，第一节点
docker create -p 13306:3306 -v haoke-v1:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e CLUSTER_NAME=pxc --name=pxc_node1 --net=pxc-network --ip=172.30.0.2 pxc

#第二节点（增加了CLUSTER_JOIN参数）
docker create -p 13307:3306 -v haoke-v2:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e CLUSTER_NAME=pxc --name=pxc_node2 -e CLUSTER_JOIN=pxc_node1 --net=pxc-network --ip=172.30.0.3 pxc

#集群2
#第一节点
docker create -p 13308:3306 -v haoke-v3:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e CLUSTER_NAME=pxc --name=pxc_node3 --net=pxc-network --ip=172.30.0.4 pxc

#第二节点（增加了CLUSTER_JOIN参数）
docker create -p 13309:3306 -v haoke-v4:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e CLUSTER_NAME=pxc --name=pxc_node4 -e CLUSTER_JOIN=pxc_node3 --net=pxc-network --ip=172.30.0.5 pxc

#启动
docker start pxc_node1 && docker logs -f pxc_node1
docker start pxc_node2 && docker logs -f pxc_node2

docker start pxc_node3 && docker logs -f pxc_node3
docker start pxc_node4 && docker logs -f pxc_node4

#查看集群节点
show status like 'wsrep_cluster%';

```

2.1.3.2、部署MS架构

```
#master
mkdir /data/mysql/haoke/master01/conf -p
vim my.cnf

```

```
#输入如下内容
[mysqld]
log-bin=mysql-bin  #开启二进制日志
server-id=1  #服务id，不可重复
sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'

#创建容器
docker create --name ms_node1 -v haoke-v5:/var/lib/mysql -v /data/mysql/haoke/master01/conf:/etc/my.cnf.d -p 13310:3306 -e MYSQL_ROOT_PASSWORD=root percona:5.7.23

#启动
docker start ms_node1 && docker logs -f ms_node1

#创建同步账户以及授权
create user 'haoke'@'%' identified by 'haoke';
grant replication slave on *.* to 'haoke'@'%';
flush privileges;

#查看master状态
show master status;
```

```
#slave
mkdir /data/mysql/haoke/slave01/conf -p
vim my.cnf

#输入如下内容
[mysqld]
server-id=2  #服务id，不可重复
sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'

#创建容器
docker create --name ms_node2 -v haoke-v6:/var/lib/mysql -v /data/mysql/haoke/slave01/conf:/etc/my.cnf.d -p 13311:3306 -e MYSQL_ROOT_PASSWORD=root percona:5.7.23

#启动
docker start ms_node2 && docker logs -f ms_node2

#设置master相关信息
CHANGE MASTER TO
 master_host='xxxxxx',
 master_user='itcast',
 master_password='itcast',
 master_port=13310,
 master_log_file='xxxxx',
 master_log_pos=xxxx;

#启动同步
start slave;
```

```
27
28 #查看master状态
29 show slave status;
```

2.1.3.3、部署mycat

在数据库中，tb_house_resources（房源表）进行pxc集群管理，其它表通过读写分离管理。

```
1  <!--房源数据表进行分片存储，切分为2个分片-->
2  <!-- server.xml -->
3
4  <?xml version="1.0" encoding="UTF-8"?>
5  <!DOCTYPE mycat:server SYSTEM "server.dtd">
6  <mycat:server xmlns:mycat="http://io.mycat/">
7      <system>
8          <property name="nonePasswordLogin">0</property>
9          <property name="useHandshakeV10">1</property>
10         <property name="useSqlStat">0</property>
11         <property name="useGlobleTableCheck">0</property>
12         <property name="sequnceHandlerType">2</property>
13         <property name="subqueryRelationshipCheck">false</property>
14         <property name="processorBufferPoolType">0</property>
15         <property name="handleDistributedTransactions">0</property>
16         <property name="useOffHeapForMerge">1</property>
17         <property name="memoryPageSize">64k</property>
18         <property name="spillsFileBufferSize">1k</property>
19         <property name="useStreamOutput">0</property>
20         <property name="systemReserveMemorySize">384m</property>
21         <property name="useZKSwitch">false</property>
22     </system>
23     <!--这里是设置的itcast用户和虚拟逻辑库-->
24     <user name="haoke" defaultAccount="true">
25         <property name="password">haoke123</property>
26         <property name="schemas">haoke</property>
27     </user>
28 </mycat:server>
29
30 <!--schema.xml-->
31
32 <?xml version="1.0"?>
33 <!DOCTYPE mycat:schema SYSTEM "schema.dtd">
34 <mycat:schema xmlns:mycat="http://io.mycat/">
35     <!--配置数据表-->
36     <schema name="haoke" checkSQLschema="false" sqlMaxLimit="100">
37         <table name="tb_house_resources" dataNode="dn1,dn2" rule="mod-long" />
38         <table name="tb_ad" dataNode="dn3"/>
39         <table name="tb_estate" dataNode="dn3"/>
40     </schema>
41
42     <!--配置分片关系-->
43     <dataNode name="dn1" dataHost="cluster1" database="haoke" />
44     <dataNode name="dn2" dataHost="cluster2" database="haoke" />
45     <dataNode name="dn3" dataHost="cluster3" database="haoke" />
```

```
    <!--配置连接信息-->
    <dataHost name="cluster1" maxCon="1000" minCon="10" balance="2"
                writeType="1" dbType="mysql" dbDriver="native" switchType="1"
                slaveThreshold="100">
        <heartbeat>select user()</heartbeat>
        <writeHost host="W1" url="192.168.1.18:13306" user="root"
                    password="root">
            <readHost host="W1R1" url="192.168.1.18:13307" user="root"
                        password="root" />
        </writeHost>
    </dataHost>
    <dataHost name="cluster2" maxCon="1000" minCon="10" balance="2"
                writeType="1" dbType="mysql" dbDriver="native" switchType="1"
                slaveThreshold="100">
        <heartbeat>select user()</heartbeat>
        <writeHost host="W2" url="192.168.1.18:13308" user="root"
                    password="root">
            <readHost host="W2R1" url="192.168.1.18:13309" user="root"
                        password="root" />
        </writeHost>
    </dataHost>
    <dataHost name="cluster3" maxCon="1000" minCon="10" balance="3"
                writeType="1" dbType="mysql" dbDriver="native" switchType="1"
                slaveThreshold="100">
        <heartbeat>select user()</heartbeat>
        <writeHost host="W2" url="192.168.1.19:13310" user="root"
                    password="root">
            <readHost host="W2R1" url="192.168.1.19:13311" user="root"
                        password="root" />
        </writeHost>
    </dataHost>
</mycat:schema>

<!-- rule.xml -->
<function name="mod-long" class="io.mycat.route.function.PartitionByMod">
    <property name="count">2</property>
</function>
```

```
#节点一
vim wrapper.conf
#设置jmx端口
wrapper.java.additional.7=-Dcom.sun.management.jmxremote.port=11986

vim server.xml
#设置服务端口以及管理端口
<property name="serverPort">18068</property>
<property name="managerPort">19068</property>

#节点二
vim wrapper.conf
#设置jmx端口
wrapper.java.additional.7=-Dcom.sun.management.jmxremote.port=11987
```

```
vim server.xml
#设置服务端口以及管理端口
<property name="serverPort">18069</property>
<property name="managerPort">19069</property>

./startup_nowrap.sh && tail -f ../logs/mycat.log
```

2.1.3.4、创建表以及测试

```
CREATE TABLE `tb_ad` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `type` int(10) DEFAULT NULL COMMENT '广告类型',
  `title` varchar(100) DEFAULT NULL COMMENT '描述',
  `url` varchar(200) DEFAULT NULL COMMENT '图片URL地址',
  `created` datetime DEFAULT NULL,
  `updated` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8 COMMENT='广告表';

CREATE TABLE `tb_estate` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) DEFAULT NULL COMMENT '楼盘名称',
  `province` varchar(10) DEFAULT NULL COMMENT '所在省',
  `city` varchar(10) DEFAULT NULL COMMENT '所在市',
  `area` varchar(10) DEFAULT NULL COMMENT '所在区',
  `address` varchar(100) DEFAULT NULL COMMENT '具体地址',
  `year` varchar(10) DEFAULT NULL COMMENT '建筑年代',
  `type` varchar(10) DEFAULT NULL COMMENT '建筑类型',
  `property_cost` varchar(10) DEFAULT NULL COMMENT '物业费',
  `property_company` varchar(20) DEFAULT NULL COMMENT '物业公司',
  `developers` varchar(20) DEFAULT NULL COMMENT '开发商',
  `created` datetime DEFAULT NULL COMMENT '创建时间',
  `updated` datetime DEFAULT NULL COMMENT '更新时间',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1006 DEFAULT CHARSET=utf8 COMMENT='楼盘表';

CREATE TABLE `tb_house_resources` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `title` varchar(100) DEFAULT NULL COMMENT '房源标题',
  `estate_id` bigint(20) DEFAULT NULL COMMENT '楼盘id',
  `building_num` varchar(5) DEFAULT NULL COMMENT '楼号（栋）',
  `building_unit` varchar(5) DEFAULT NULL COMMENT '单元号',
  `building_floor_num` varchar(5) DEFAULT NULL COMMENT '门牌号',
  `rent` int(10) DEFAULT NULL COMMENT '租金',
  `rent_method` tinyint(1) DEFAULT NULL COMMENT '租赁方式，1-整租，2-合租',
  `payment_method` tinyint(1) DEFAULT NULL COMMENT '支付方式，1-付一押一，2-付三押一，3-付六押一，4-年付押一，5-其它',
  `house_type` varchar(255) DEFAULT NULL COMMENT '户型，如：2室1厅1卫',
  `covered_area` varchar(10) DEFAULT NULL COMMENT '建筑面积',
  `use_area` varchar(10) DEFAULT NULL COMMENT '使用面积',
  `floor` varchar(10) DEFAULT NULL COMMENT '楼层，如：8/26',
  `orientation` varchar(2) DEFAULT NULL COMMENT '朝向：东、南、西、北',
```

```
  `decoration` tinyint(1) DEFAULT NULL COMMENT '装修，1-精装，2-简装，3-毛坯',
  `facilities` varchar(50) DEFAULT NULL COMMENT '配套设施， 如：1,2,3',
  `pic` varchar(1000) DEFAULT NULL COMMENT '图片，最多5张',
  `house_desc` varchar(200) DEFAULT NULL COMMENT '描述',
  `contact` varchar(10) DEFAULT NULL COMMENT '联系人',
  `mobile` varchar(11) DEFAULT NULL COMMENT '手机号',
  `time` tinyint(1) DEFAULT NULL COMMENT '看房时间，1-上午，2-中午，3-下午，4-晚上，5-全天',
  `property_cost` varchar(10) DEFAULT NULL COMMENT '物业费',
  `created` datetime DEFAULT NULL,
  `updated` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8 COMMENT='房源表';


INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('1', '1', 'UniCity万科天空之城', 'http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/26/15432029097062227.jpg', '2018-11-26 11:28:49', '2018-11-26 11:28:51');
INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('2', '1', '天和尚海庭前', 'http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/26/1543202958579877.jpg', '2018-11-26 11:29:27', '2018-11-26 11:29:29');
INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('3', '1', '[奉贤 南桥] 光语著', 'http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/26/15432029946721854.jpg', '2018-11-26 11:30:04', '2018-11-26 11:30:06');
INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('4', '1', '[上海周边 嘉兴] 融创海逸长洲', 'http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/26/15432030275359146.jpg', '2018-11-26 11:30:49', '2018-11-26 11:30:53');


INSERT INTO `tb_estate` (`id`, `name`, `province`, `city`, `area`, `address`, `year`, `type`, `property_cost`, `property_company`, `developers`, `created`, `updated`) VALUES ('1001', '中远两湾城', '上海市', '上海市', '普陀区', '远景路97弄', '2001', '塔楼/板楼', '1.5', '上海中远物业管理发展有限公司', '上海万业企业股份有限公司', '2018-11-06 23:00:20', '2018-11-06 23:00:23');
INSERT INTO `tb_estate` (`id`, `name`, `province`, `city`, `area`, `address`, `year`, `type`, `property_cost`, `property_company`, `developers`, `created`, `updated`) VALUES ('1002', '上海康城', '上海市', '上海市', '闵行区', '莘松路958弄', '2001', '塔楼/板楼', '1.5', '盛孚物业', '闵行房地产', '2018-11-06 23:02:30', '2018-11-27 23:02:33');
INSERT INTO `tb_estate` (`id`, `name`, `province`, `city`, `area`, `address`, `year`, `type`, `property_cost`, `property_company`, `developers`, `created`, `updated`) VALUES ('1003', '保利西子湾', '上海市', '上海市', '松江区', '广富林路1188弄', '2008', '塔楼/板楼', '1.75', '上海保利物业管理', '上海城乾房地产开发有限公司', '2018-11-06 23:04:22', '2018-11-06 23:04:25');
INSERT INTO `tb_estate` (`id`, `name`, `province`, `city`, `area`, `address`, `year`, `type`, `property_cost`, `property_company`, `developers`, `created`, `updated`) VALUES ('1004', '万科城市花园', '上海市', '上海市', '松江区', '广富林路1188弄', '2002', '塔楼/板楼', '1.5', '上海保利物业管理', '上海城乾房地产开发有限公司', '2018-11-13 16:43:40', '2018-11-13 16:43:42');
```

```
67 INSERT INTO `tb_estate` (`id`, `name`, `province`, `city`, `area`, `address`, `year`,
   `type`, `property_cost`, `property_company`, `developers`, `created`, `updated`)
   VALUES ('1005', '上海阳城', '上海市', '上海市', '闵行区', '罗锦路888弄', '2002', '塔楼/板楼',
   '1.5', '上海莲阳物业管理有限公司', '上海莲城房地产开发有限公司', '2018-11-06 23:23:52', '2018-
   11-06 23:23:55');
68
69
70 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`,
   `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`,
   `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`,
   `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`,
   `created`, `updated`) VALUES ('1', '东方曼哈顿 3室2厅 16000元', '1005', '2', '1', '1',
   '1111', '1', '1', '1室1厅1卫1厨1阳台', '2', '2', '1/2', '南', '1', '1,2,3,8,9', NULL, '这
   个经纪人很懒，没写核心卖点', '张三', '11111111111', '1', '11', '2018-11-16 01:16:00',
   '2018-11-16 01:16:00');
71 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`,
   `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`,
   `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`,
   `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`,
   `created`, `updated`) VALUES ('2', '康城 3室2厅1卫', '1002', '1', '2', '3', '2000', '1',
   '2', '3室2厅1卫1厨2阳台', '100', '80', '2/20', '南', '1', '1,2,3,7,6', NULL, '拎包入住',
   '张三', '18888888888', '5', '1.5', '2018-11-16 01:34:02', '2018-11-16 01:34:02');
72 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`,
   `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`,
   `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`,
   `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`,
   `created`, `updated`) VALUES ('3', '2', '1002', '2', '2', '2', '2', '1', '1', '1室1厅1
   卫1厨1阳台', '22', '11', '1/5', '南', '1', '1,2,3', NULL, '11', '22', '33', '1', '3',
   '2018-11-16 21:15:29', '2018-11-16 21:15:29');
73 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`,
   `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`,
   `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`,
   `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`,
   `created`, `updated`) VALUES ('4', '11', '1002', '1', '1', '1', '1', '1', '1', '1室1厅1
   卫1厨1阳台', '11', '1', '1/1', '南', '1', '1,2,3', NULL, '11', '1', '1', '1', '1',
   '2018-11-16 21:16:50', '2018-11-16 21:16:50');
74 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`,
   `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`,
   `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`,
   `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`,
   `created`, `updated`) VALUES ('5', '最新修改房源5', '1002', '1', '1', '1', '3000', '1',
   '1', '1室1厅1卫1厨1阳台', '80', '1', '1/1', '南', '1', '1,2,3', 'http://itcast-haoke.oss-
   cn-qingdao.aliyuncs.com/images/2018/12/04/15439353467987363.jpg,http://itcast-
   haoke.oss-cn-qingdao.aliyuncs.com/images/2018/12/04/15439354795233043.jpg', '11', '1',
   '1', '1', '1', '2018-11-16 21:17:02', '2018-12-04 23:05:19');
```

```
75 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`, `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`, `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`, `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`, `created`, `updated`) VALUES ('6', '房源标题', '1002', '1', '1', '11', '1', '1', '1', '1室1厅1卫1厨1阳台', '11', '1', '1/1', '南', '1', '1,2,3', 'http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/16/15423743004743329.jpg,http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/16/15423743049233737.jpg', '11', '2', '2', '1', '1', '2018-11-16 21:18:41', '2018-11-16 21:18:41');
76 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`, `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`, `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`, `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`, `created`, `updated`) VALUES ('7', '房源标题', '1002', '1', '1', '11', '1', '1', '1', '1室1厅1卫1厨1阳台', '11', '1', '1/1', '南', '1', '1,2,3', 'http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/16/15423743004743329.jpg,http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/16/15423743049233737.jpg', '11', '2', '2', '1', '1', '2018-11-16 21:18:41', '2018-11-16 21:18:41');
77 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`, `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`, `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`, `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`, `created`, `updated`) VALUES ('8', '3333', '1002', '1', '1', '1', '1', '1', '1', '1室1厅1卫1厨1阳台', '1', '1', '1/1', '南', '1', '1,2,3', 'http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/17/15423896060254118.jpg,http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/17/15423896084306516.jpg', '1', '1', '1', '1', '1', '2018-11-17 01:33:35', '2018-12-06 10:22:20');
78 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`, `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`, `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`, `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`, `created`, `updated`) VALUES ('9', '康城 精品房源2', '1002', '1', '2', '3', '1000', '1', '1', '1室1厅1卫1厨1阳台', '50', '40', '3/20', '南', '1', '1,2,3', 'http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/30/15435106627858721.jpg,http://itcast-haoke.oss-cn-qingdao.aliyuncs.com/images/2018/11/30/15435107119124432.jpg', '精品房源', '李四', '18888888888', '1', '1', '2018-11-21 18:31:35', '2018-11-30 00:58:46');
```

2.1.3.5、部署HAProxy

```
1 #拉取镜像
2 docker pull haproxy:1.9.3
3
4 #创建目录，用于存放配置文件
5 mkdir /haoke/haproxy
6
7 #创建容器
8 docker create --name haproxy --net host -v /haoke/haproxy:/usr/local/etc/haproxy haproxy:1.9.3
```

编写配置文件：

```
#创建文件
vim /haoke/haproxy/haproxy.cfg

#输入如下内容
global
    log         127.0.0.1 local2
    maxconn     4000
    daemon

defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

listen   admin_stats
    bind    0.0.0.0:4001
    mode  http
    stats uri       /dbs
    stats realm  Global\ statistics
    stats auth    admin:admin123

listen   proxy-mysql
    bind    0.0.0.0:4002
    mode  tcp
    balance  roundrobin
    option   tcplog
    #代理mycat服务
    server   mycat_1  192.168.1.19:18068  check  port  18068  maxconn  2000
    server   mycat_2  192.168.1.19:18069  check  port  18069  maxconn  2000
```

启动容器：

```
#启动容器
docker restart haproxy && docker logs -f haproxy
```

2.2、部署Redis集群

Redis集群采用3主3从的架构。

2.2.1、规划

服务	端口	服务器	容器名
Redis-node01	6379	192.168.1.18	redis-node01
Redis-node02	6380	192.168.1.18	redis-node02
Redis-node03	6381	192.168.1.18	redis-node03
Redis-node04	16379	192.168.1.19	redis-node04
Redis-node05	16380	192.168.1.19	redis-node05
Redis-node06	16381	192.168.1.19	redis-node06

2.2.3、实施

```
docker volume create redis-node01
docker volume create redis-node02
docker volume create redis-node03
docker volume create redis-node04
docker volume create redis-node05
docker volume create redis-node06

#创建容器
docker create --name redis-node01 --net host -v redis-node01:/data redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-01.conf --port 6379

docker create --name redis-node02 --net host -v redis-node02:/data redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-02.conf --port 6380

docker create --name redis-node03 --net host -v redis-node03:/data redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-03.conf --port 6381

docker create --name redis-node04 --net host -v redis-node04:/data redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-03.conf --port 16379

docker create --name redis-node05 --net host -v redis-node05:/data redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-03.conf --port 16380

docker create --name redis-node06 --net host -v redis-node06:/data redis:5.0.2 --cluster-enabled yes --cluster-config-file nodes-node-03.conf --port 16381

#启动容器
docker start redis-node01 redis-node02 redis-node03 redis-node04 redis-node05 redis-node06

#进入redis-node01容器进行操作
docker exec -it redis-node01 /bin/bash

#192.168.1.18，192.168.1.19是主机的ip地址
```

```
redis-cli --cluster create 192.168.1.18:6379 192.168.1.18:6380 192.168.1.18:6381
192.168.1.19:16379 192.168.1.19:16380 192.168.1.19:16381 --cluster-replicas 1
```

2.3、部署Elasticsearch集群

Elasticsearch集群部署3个节点的集群。

2.3.1、规划

服务	端口	服务器	容器名
es-node01	9200，9300	192.168.1.7	es-node01
es-node02	9200，9300	192.168.1.18	es-node02
es-node03	9200，9300	192.168.1.19	es-node03

2.3.2、实施

```

#elasticsearch.yml：
cluster.name: es-haoke-cluster
node.name: node01
node.master: true
node.data: true
network.host: 192.168.1.7
http.port: 9200
discovery.zen.ping.unicast.hosts: ["192.168.1.7","192.168.1.18","192.168.1.19"]
discovery.zen.minimum_master_nodes: 2
http.cors.enabled: true
http.cors.allow-origin: "*"

#jvm.options
-Xms512m
-Xmx512m

#将IK的zip压缩包解压到/haoke/es-cluster/ik

docker create --name es-node01 --net host -v /haoke/es-
cluster/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml -v
/haoke/es-cluster/jvm.options:/usr/share/elasticsearch/config/jvm.options -v
/haoke/es-cluster/data:/usr/share/elasticsearch/data -v /haoke/es-
cluster/ik:/usr/share/elasticsearch/plugins/ik -v /haoke/es-
cluster/pinyin:/usr/share/elasticsearch/plugins/pinyin elasticsearch:6.5.4

docker create --name es-node02 --net host -v /haoke/es-
cluster/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml -v
/haoke/es-cluster/jvm.options:/usr/share/elasticsearch/config/jvm.options -v
/haoke/es-cluster/data:/usr/share/elasticsearch/data -v /haoke/es-
cluster/ik:/usr/share/elasticsearch/plugins/ik -v /haoke/es-
cluster/pinyin:/usr/share/elasticsearch/plugins/pinyin elasticsearch:6.5.4

```

```
docker create --name es-node03 --net host -v /haoke/es-cluster/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml -v /haoke/es-cluster/jvm.options:/usr/share/elasticsearch/config/jvm.options -v /haoke/es-cluster/data:/usr/share/elasticsearch/data -v /haoke/es-cluster/ik:/usr/share/elasticsearch/plugins/ik -v /haoke/es-cluster/pinyin:/usr/share/elasticsearch/plugins/pinyin elasticsearch:6.5.4

#启动测试
docker start es-node01 && docker logs -f es-node01

docker start es-node02 && docker logs -f es-node02

docker start es-node03 && docker logs -f es-node03
```

2.3.3、文档mapping

```
PUT http://192.168.1.7:9200/haoke/

{
    "settings": {
        "index": {
            "number_of_shards": 6,
            "number_of_replicas": 1,
        "analysis": {
                "analyzer": {
                    "pinyin_analyzer": {
                        "tokenizer": "my_pinyin"
                    }
                },
                "tokenizer": {
                    "my_pinyin": {
                        "type": "pinyin",
                        "keep_separate_first_letter": false,
                        "keep_full_pinyin": true,
                        "keep_original": true,
                        "limit_first_letter_length": 16,
                        "lowercase": true,
                        "remove_duplicated_term": true
                    }
                }
            }
        }
    },
    "mappings": {
        "house": {
            "dynamic": false,
            "properties": {
                "title": {
                    "type": "text",
                    "analyzer":"ik_max_word",
                    "fields":{
                        "pinyin":{
                            "type": "text",
```

```
                            "analyzer": "pinyin_analyzer"
                        }
                    }
                },
                "image": {
                    "type": "keyword",
                    "index":false
                },
                "orientation": {
                    "type": "keyword",
                    "index":false
                },
                "houseType": {
                    "type": "keyword",
                    "index":false
                },
                "rentMethod": {
                    "type": "keyword",
                    "index":false
                },
                "time": {
                    "type": "keyword",
                    "index":false
                },
                "rent": {
                    "type": "keyword",
                    "index":false
                },
                "floor": {
                    "type": "keyword",
                    "index":false
                }
            }
        }
    }
}

```

2.3.4、导入数据

```
@Test
    public void tesBulk() throws Exception {
        Request request = new Request("POST", "/haoke/house/_bulk");
        List<String> lines = FileUtils.readLines(new File("F:\\code\\data.json"),
"UTF-8");
        String createStr = "{\"index\":{\"_index\":\"haoke\",\"_type\":\"house\"}}";
        StringBuilder sb = new StringBuilder();
        int count = 0;
        for (String line : lines) {

            sb.append(createStr + "\n" + line + "\n");

            if (count >= 100) {
```

```
                request.setJsonEntity(sb.toString());
                Response response = this.restClient.performRequest(request);
                System.out.println("请求完成 -> " + response.getStatusLine());
                System.out.println(EntityUtils.toString(response.getEntity()));

                count = 0;
                sb = new StringBuilder();
            }
            count++;

        }

    }
```

2.4、部署RocketMQ集群

搭建2master+2slave的集群。

2.4.1、规划

服务	端口	服务器	容器名
rmqserver01	9876	192.168.1.7	rmqserver01
rmqserver02	9877	192.168.1.7	rmqserver02
rmqbroker01	10911	192.168.1.19	rmqbroker01
rmqbroker02	10811	192.168.1.19	rmqbroker02
rmqbroker01-slave	10711	192.168.1.18	rmqbroker01-slave
rmqbroker02-slave	10611	192.168.1.18	rmqbroker02-slave
rocketmq-console	8082	192.168.1.7	rocketmq-console

2.4.2、实施

```
#创建2个master
#nameserver1
docker create -p 9876:9876 --name rmqserver01 \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-e "JAVA_OPTS=-Duser.home=/opt"  \
-v /haoke/rmq/rmqserver01/logs:/opt/logs \
-v /haoke/rmq/rmqserver01/store:/opt/store \
foxiswho/rocketmq:server-4.3.2

#nameserver2
docker create -p 9877:9876 --name rmqserver02 \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-e "JAVA_OPTS=-Duser.home=/opt"  \
-v /haoke/rmq/rmqserver02/logs:/opt/logs \
```

```
-v /haoke/rmq/rmqserver02/store:/opt/store \
foxiswho/rocketmq:server-4.3.2
```

```
#创建第1个master broker
#master broker01
docker create --net host --name rmqbroker01 \
-e "JAVA_OPTS=-Duser.home=/opt"  \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-v /haoke/rmq/rmqbroker01/conf/broker.conf:/etc/rocketmq/broker.conf \
-v /haoke/rmq/rmqbroker01/logs:/opt/logs \
-v /haoke/rmq/rmqbroker01/store:/opt/store \
foxiswho/rocketmq:broker-4.3.2

#配置
namesrvAddr=192.168.1.7:9876;192.168.1.7:9877
brokerClusterName=haokeCluster
brokerName=broker01
brokerId=0
deleteWhen=04
fileReservedTime=48
brokerRole=SYNC_MASTER
flushDiskType=ASYNC_FLUSH
brokerIP1=192.168.1.19
brokerIp2=192.168.1.19
listenPort=10911
```

```
#创建第2个master broker
#master broker02
docker create --net host --name rmqbroker02 \
-e "JAVA_OPTS=-Duser.home=/opt"  \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-v /haoke/rmq/rmqbroker02/conf/broker.conf:/etc/rocketmq/broker.conf \
-v /haoke/rmq/rmqbroker02/logs:/opt/logs \
-v /haoke/rmq/rmqbroker02/store:/opt/store \
foxiswho/rocketmq:broker-4.3.2

#master broker02
namesrvAddr=192.168.1.7:9876;192.168.1.7:9877
brokerClusterName=haokeCluster
brokerName=broker02
brokerId=0
deleteWhen=04
fileReservedTime=48
brokerRole=SYNC_MASTER
flushDiskType=ASYNC_FLUSH
brokerIP1=192.168.1.19
brokerIp2=192.168.1.19
listenPort=10811
```

```
#创建第1个slave broker
#slave broker01
```

```
docker create --net host --name rmqbroker03 \
-e "JAVA_OPTS=-Duser.home=/opt"  \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-v /haoke/rmq/rmqbroker03/conf/broker.conf:/etc/rocketmq/broker.conf \
-v /haoke/rmq/rmqbroker03/logs:/opt/logs \
-v /haoke/rmq/rmqbroker03/store:/opt/store \
foxiswho/rocketmq:broker-4.3.2

#slave broker01
namesrvAddr=192.168.1.7:9876;192.168.1.7:9877
brokerClusterName=haokeCluster
brokerName=broker01
brokerId=1
deleteWhen=04
fileReservedTime=48
brokerRole=SLAVE
flushDiskType=ASYNC_FLUSH
brokerIP1=192.168.1.18
brokerIp2=192.168.1.18
listenPort=10711
```

```
#创建第2个slave broker
#slave broker01
docker create --net host --name rmqbroker04 \
-e "JAVA_OPTS=-Duser.home=/opt"  \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-v /haoke/rmq/rmqbroker04/conf/broker.conf:/etc/rocketmq/broker.conf \
-v /haoke/rmq/rmqbroker04/logs:/opt/logs \
-v /haoke/rmq/rmqbroker04/store:/opt/store \
foxiswho/rocketmq:broker-4.3.2

#slave broker02
namesrvAddr=192.168.1.7:9876;192.168.1.7:9877
brokerClusterName=haokeCluster
brokerName=broker02
brokerId=1
deleteWhen=04
fileReservedTime=48
brokerRole=SLAVE
flushDiskType=ASYNC_FLUSH
brokerIP1=192.168.1.18
brokerIp2=192.168.1.18
listenPort=10611
```

```
#启动容器
docker start rmqserver01 rmqserver02
docker start rmqbroker01 rmqbroker02
docker start rmqbroker03 rmqbroker04
```

```
#rocketmq-console的部署安装
#拉取镜像
docker pull styletang/rocketmq-console-ng:1.0.0

#创建并启动容器
docker run -e "JAVA_OPTS=-Drocketmq.namesrv.addr=192.168.1.7:9876;192.168.1.7:9877 -Dcom.rocketmq.sendMessageWithVIPChannel=false" -p 8082:8080 -t styletang/rocketmq-console-ng:1.0.0
```

2.5、搭建ZK集群

搭建3个节点的zk集群。

2.5.1、规划

服务	端口	服务器	容器名
zk01	2181，2888，3888	192.168.1.7	zk01
zk02	2181，2888，3888	192.168.1.18	zk02
zk03	2181，2888，3888	192.168.1.19	zk03

2.5.2、实施

```
#拉取zk镜像
docker pull zookeeper:3.4

#创建容器
docker create --name zk01 --net host -e ZOO_MY_ID=1 -e ZOO_SERVERS="server.1=192.168.1.7:2888:3888 server.2=192.168.1.18:2888:3888 server.3=192.168.1.19:2888:3888" zookeeper:3.4

docker create --name zk02 --net host -e ZOO_MY_ID=2 -e ZOO_SERVERS="server.1=192.168.1.7:2888:3888 server.2=192.168.1.18:2888:3888 server.3=192.168.1.19:2888:3888" zookeeper:3.4

docker create --name zk03 --net host -e ZOO_MY_ID=3 -e ZOO_SERVERS="server.1=192.168.1.7:2888:3888 server.2=192.168.1.18:2888:3888 server.3=192.168.1.19:2888:3888" zookeeper:3.4

#启动容器
docker start zk01 && docker logs -f zk01
docker start zk02 && docker logs -f zk02
docker start zk03 && docker logs -f zk03
```

3、项目打包

项目的域名规划（虚拟域名）：

项目	域名	机器
itcast-haoke-manage-api-server	api.manage.haoke.com	192.168.1.7
itcast-haoke-manage-web	manage.haoke.com	192.168.1.7
itcast-haoke-web	www.haoke.com	192.168.1.18
itcast-haoke-im	im.haoke.com	192.168.1.19

3.1、打包springboot项目

第一步：添加springboot的打包插件

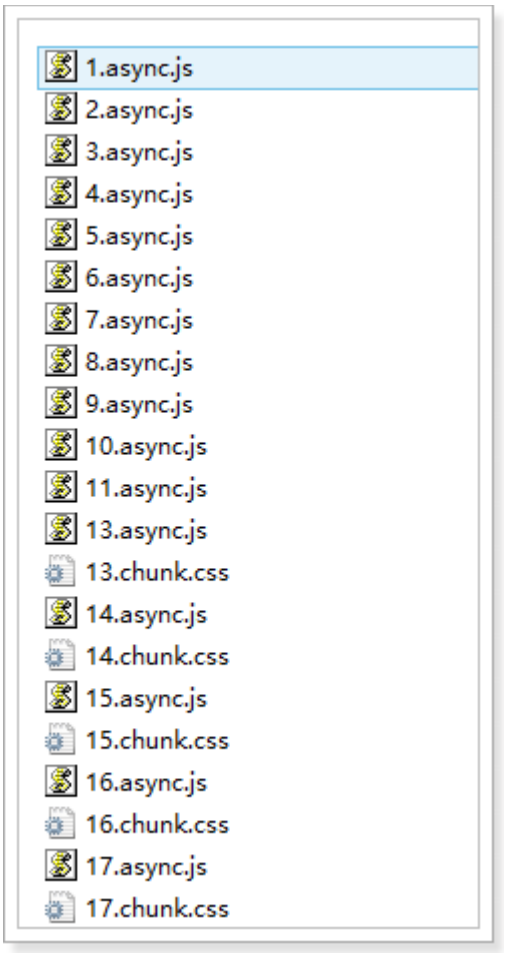
```
<build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <!--替换成实际的类全路径-->
                    <mainClass>cn.itcast.haoke.dubbo.server.DubboProvider</mainClass>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

第二步：执行打包命令

```
mvn install -Dmaven.test.skip=true
```

3.2、构建Ant Design Pro

```
#通过umi命令进行构建，构建成功后会生成静态页面
umi build
```



生成的静态页面需要通过nginx进行访问，并且请求数据的代理也需要通过nginx进行代理。

```
#安装nginx
apt install libpcre3 libpcre3-dev zlib1g-dev openssl libssl-dev

./configure
make install
#启动
cd /usr/local/nginx/sbin/
./nginx
```

3.3、itcast-haoke-manage-web系统的nginx配置

```
    server {
        listen       80;
        server_name  manage.haoke.com;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

```

```
        location ^~ /haoke/ {
            proxy_pass http://192.168.1.7:18080/;
            proxy_connect_timeout 600;
            proxy_read_timeout 600;
        }

        location / {
            root   /haoke/publish/manage-web;
        }

    }
```

3.4、配置虚拟域名

```
    server {
        listen       80;
        server_name  api.manage.haoke.com;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

    proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        location / {
            proxy_pass http://192.168.1.7:18080;
            proxy_connect_timeout 600;
            proxy_read_timeout 600;
        }

    }

```

3.5、nginx反向代理websocket

```
    server {
        listen       80;
        server_name  im.haoke.com;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        location / {
                proxy_pass http://192.168.1.19:18081;

```

```
                proxy_connect_timeout 600;
                proxy_read_timeout 600;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
        }

    }
```