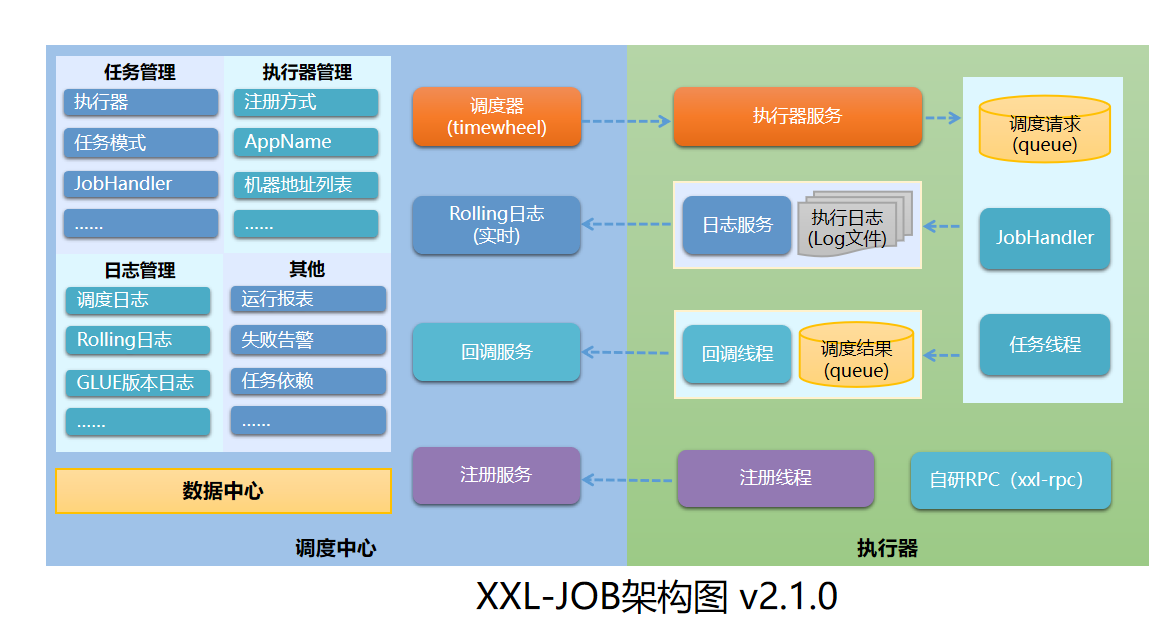
Xxl-job

**1.xxl-job介绍：**

Xxl-job是一个分布式定时任务的调度器。传统的schedule定时任务在微服务多节点的时候同一个任务会被多个节点同时运行，导致冲突从而产生问题，而xxl-job使用调度中心对多个节点的相同任务进行统一的调度，保证在同一个时候相同任务不会被重复调用。

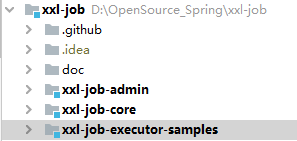
**2.xxl-job架构图**



Xxl-job的设计理念就是将任务的调度和任务运行进行解耦。

Xxl-job主要分成两个部分：

代码结构截图如下：



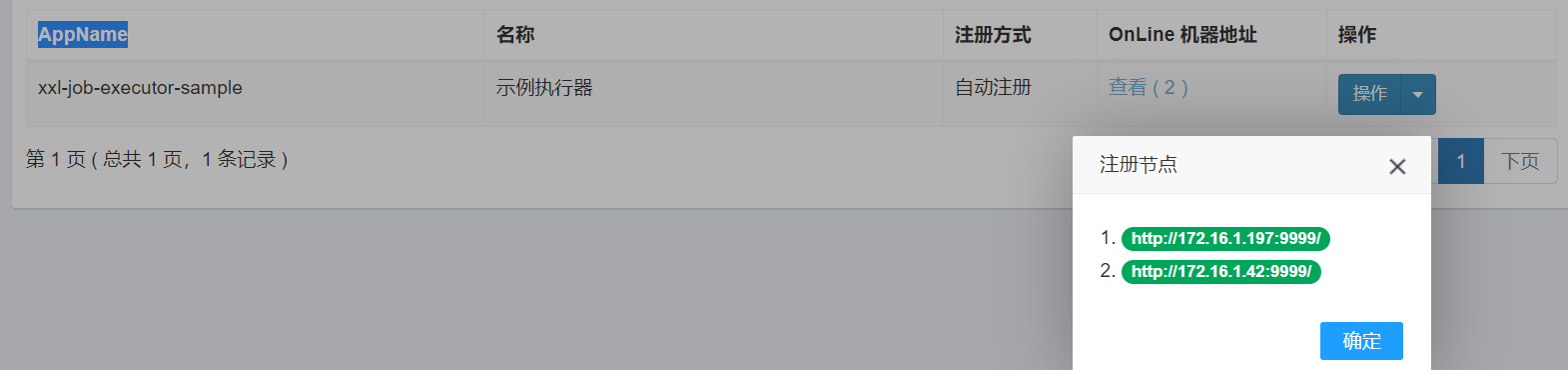
Xxl-job使用的是springboot实现的，xxl-job-admin就是调度中心部分，xxl-job-core是调度中心和执行器共同调用的核心库，xxl-job-executor-samples是执行器的示例。

**调度中心：负责所有任务的管理和调度。**

**执行器的管理：**

一个执行器组可以包含多个需要调度功能的服务执行器节点，微服务将自己的执行器注册到执行器组管理器中。

如下图所示就是一个包含两个执行器(一般是两个相同的微服务节点)的执行器组。



**任务的管理：**

定时任务调度管理，一个执行器组可以包含多个定时任务，任务的调度方式有以下方式：

1.FIRST（第一个）：固定选择第一个机器；

2.LAST（最后一个）：固定选择最后一个机器；

3.ROUND（轮询）：；

4.RANDOM（随机）：随机选择在线的机器；

5.CONSISTENT\_HASH（一致性HASH）：每个任务按照Hash算法固定选择某一台机器，且所有任务均匀散列在不同机器上。

6.LEAST\_FREQUENTLY\_USED（最不经常使用）：使用频率最低的机器优先被选举；

7.LEAST\_RECENTLY\_USED（最近最久未使用）：最久未使用的机器优先被选举；

8.FAILOVER（故障转移）：按照顺序依次进行心跳检测，第一个心跳检测成功的机器选定为目标执行器并发起调度；

9.BUSYOVER（忙碌转移）：按照顺序依次进行空闲检测，第一个空闲检测成功的机器选定为目标执行器并发起调度；

10.SHARDING\_BROADCAST(分片广播)：广播触发对应集群中所有机器执行一次任务，同时系统自动传递分片参数；可根据分片参数开发分片任务；

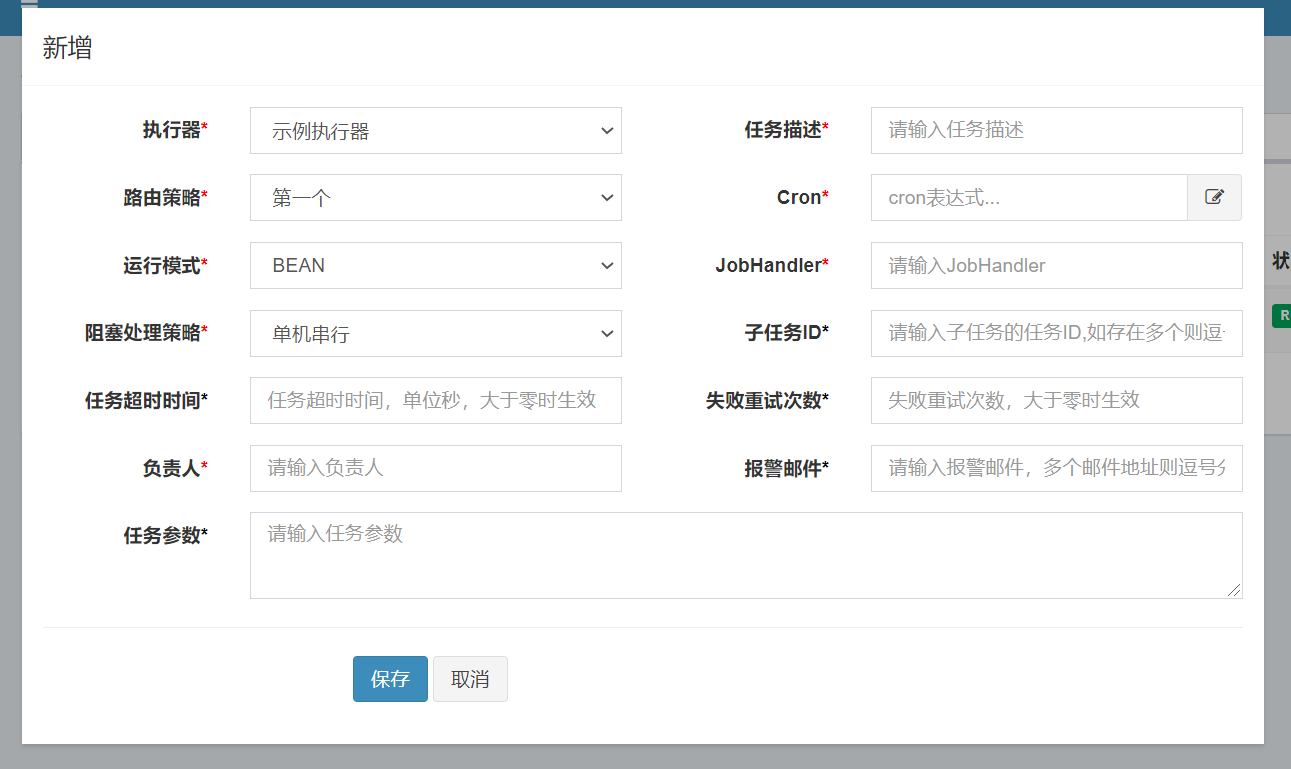
任务的运行模式：

BEAN模式:

需要在微服务节点中提前实现业务逻辑的定时任务，使用jobhandler和服务节点中的任务进行关联。

GLUE模式:

任务以源码方式维护在调度中心，支持通过Web IDE在线更新，实时编译和生效，因此不需要指定JobHandler。



这样通过对执行器中任务的统一调度管理就不会出现同一个任务在不同节点上同时运行的问题，当然如果想要同一个任务在执行器组的所有节点上运行可以使用SHARDING\_BROADCAST路由模式来运行任务。

**执行器：具体业务逻辑任务的运行**

执行器就是我们项目中所要运行的业务逻辑定时任务的一个组，一般集成在我们项目的服务中。

**3.xxl-job部署和使用**

由于调度中心和执行器中都有很多的服务在相互协作运行，所以调度中心和执行器必须在一个能相互ping通的环境中运行。一种方式是调度中心和执行器都运行在本地，还有一种是云端搭建一个调度中心和执行器相互协作的运行环境。

目前我们研究的时候使用的10.16.33.51/10.16.33.73/10.16.35.160这三台台机器搭的一个运行环境。后面在开发中对定时任务调试的过程中，最好使用xxl-job-admin项目代码在本地运行一个调度中心桩进行调试。

**3.1 调度中心部署(10.16.33.73)：**

地址：http://10.16.33.73:8282/xxl-job-admin

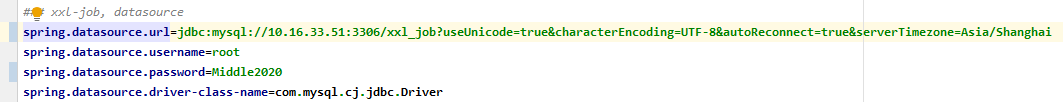
用户名/密码：admin/123456

3.1.1使用\xxl-job\doc\db\tables\_xxl\_job.sql脚本将调度中心的数据中心部署到10.16.33.51数据库服务器上。

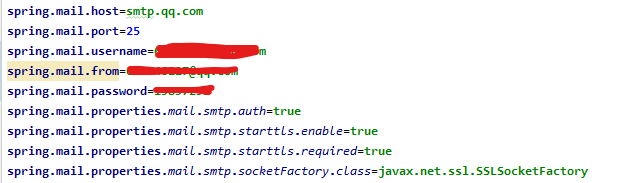
3.1.2 配置调度中心项目中的属性

xxl-job\xxl-job-admin\src\main\resources\application.properties

数据中心地址：



报警邮箱配置：



3.1.3 将代码拷贝到10.16.33.73:/data/jenkins/workspace/xxl-job-dispatch-center 下面

进入jenkins： <http://10.16.33.73:6060/job/xxl-job-dispatch-center/> 对调度中心进行构建同时部署到10.16.33.73，打开http://10.16.33.73:8282/xxl-job-admin进行登录。

至此调度中心部署成功。

**3.2执行器的创建和部署（**10.16.33.73/10.16.35.160**）**

使用xxl-job项目中的xxl-job-executor-sample-springboot项目作为执行器的示例项目。

3.2.1 执行器的配置

\xxl-job\xxl-job-executor-samples\xxl-job-executor-sample-springboot\src\main\

resources\application.properties

修改调度中心的地址：



\xxl-job\xxl-job-executor-samples\xxl-job-executor-sample-springboot\src\main\java\com\xxl\job\exe cutor\core\config\XxlJobConfig.java

使用interface ip作为执行器的注册地址：



3.2.2 在调度中心新建一个和xxl.job.executor.appname

配置一样名称的执行器组。

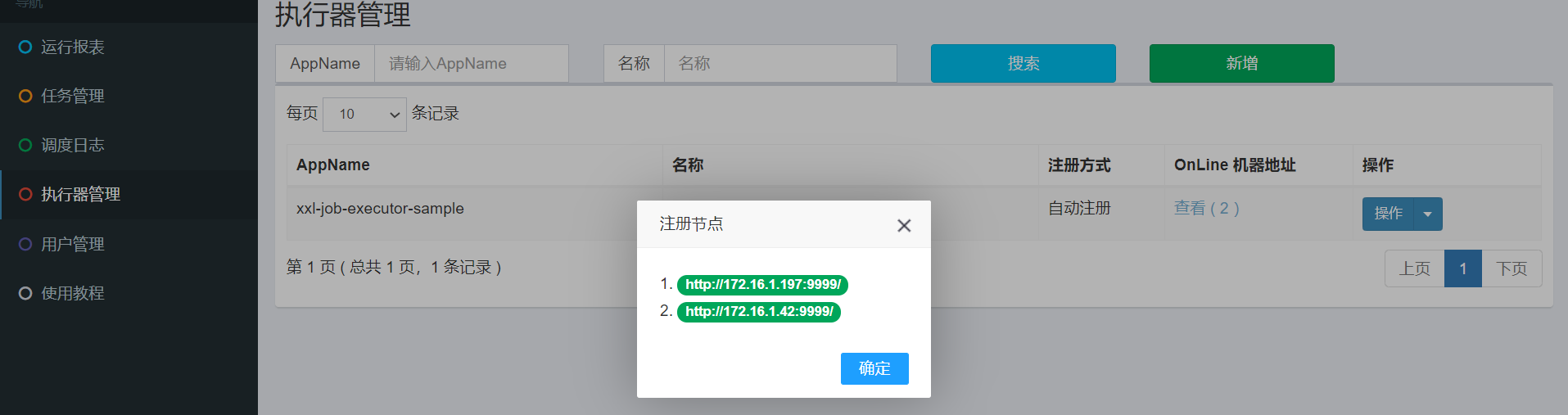
如下图所示：



3.2.3 将代码拷贝到10.16.33.73:/data/jenkins/workspace/xxl-job-executor-sample 下面

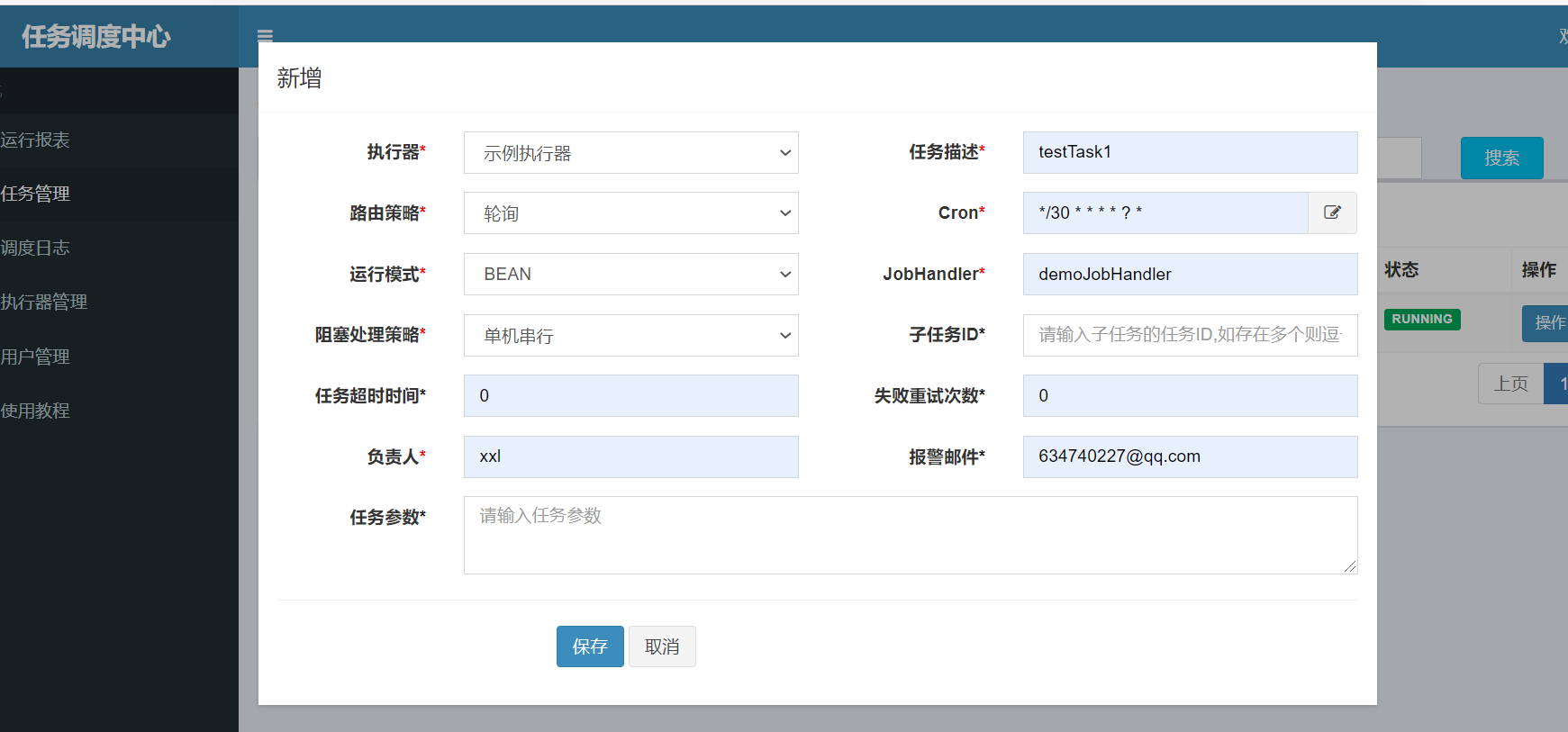
进入jenkins： <http://10.16.33.73:6060/job/xxl-job-executor-sample> 对执行器项目进行构建同时部署到10.16.33.73和10.16.35.160。

这样10.16.33.73和10.16.35.160两个执行器都会注册到xxl-job-executor-sample这个执行器组下，如图所示：

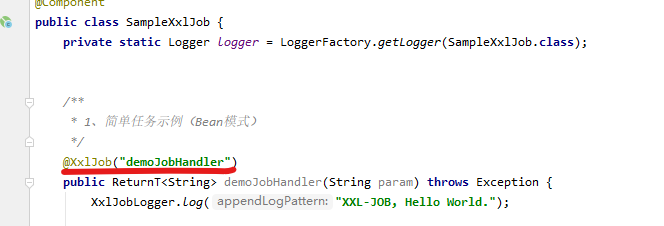


172.16.1.197和172.16.1.42分别对应10.16.33.73和10.16.35.160两台虚拟服务器的虚拟网卡的地址。可以进入服务器使用ifconfig命令查看。

3.2.4 为执行器在调度中心创建对应的bean模式的任务如下图所示：



demoJobHandler就是对应执行器项目中\xxl-job-executor-sample-springboot\src\main\java\com\xxl\job\executor\service\jobhandler\SampleXxlJob.java



Cron = \*/30 \* \* \* \* ? \* 让demoJobHandler这个任务30秒执行一次。

任务创建完成以后执行任务。

3.2.5 进入调度日志查看调度状况，或者进入10.16.33.73和10.16.35.160：/data/applogs/xxl-job查看执行器服务运行的log：xxl-job-executor-sample-springboot.log查看任务运行情况。

