

Alibaba微服务组件Nacos注册中心

1. 什么是 Nacos

2. Nacos注册中心

2.1 注册中心演变及其设计思想

2.2 Nacos注册中心架构

2.3 核心功能

3 Nacos Server部署

3.1 单机模式

3.2 集群模式

1.3 prometheus+grafana监控Nacos

4. Spring Cloud Alibaba Nacos快速开始

4.1 Spring Cloud Alibaba版本选型

4.2 搭建Nacos-client服务

1. 什么是 Nacos

官方：一个更易于构建云原生应用的动态**服务发现**([Nacos Discovery](#))、**服务配置**([Nacos Config](#))和服务管理平台。

集 注册中心+配置中心+服务管理 平台

Nacos 的关键特性包括:

- 服务发现和服务健康监测
- 动态配置服务
- 动态 DNS 服务
- 服务及其元数据管理

2. Nacos注册中心

管理所有微服务、解决微服务之间调用关系错综复杂、难以维护的问题;

2.1 注册中心演变及其设计思想

```
String url = "http://localhost:8010/order/findOrderByUserId/"+id;  
ResponseEntity<List> responseEntity = restTemplate.getForEntity(url,List.class);  
List<Order> orderList = responseEntity.getBody();
```

RestTemplate

Http远程调用

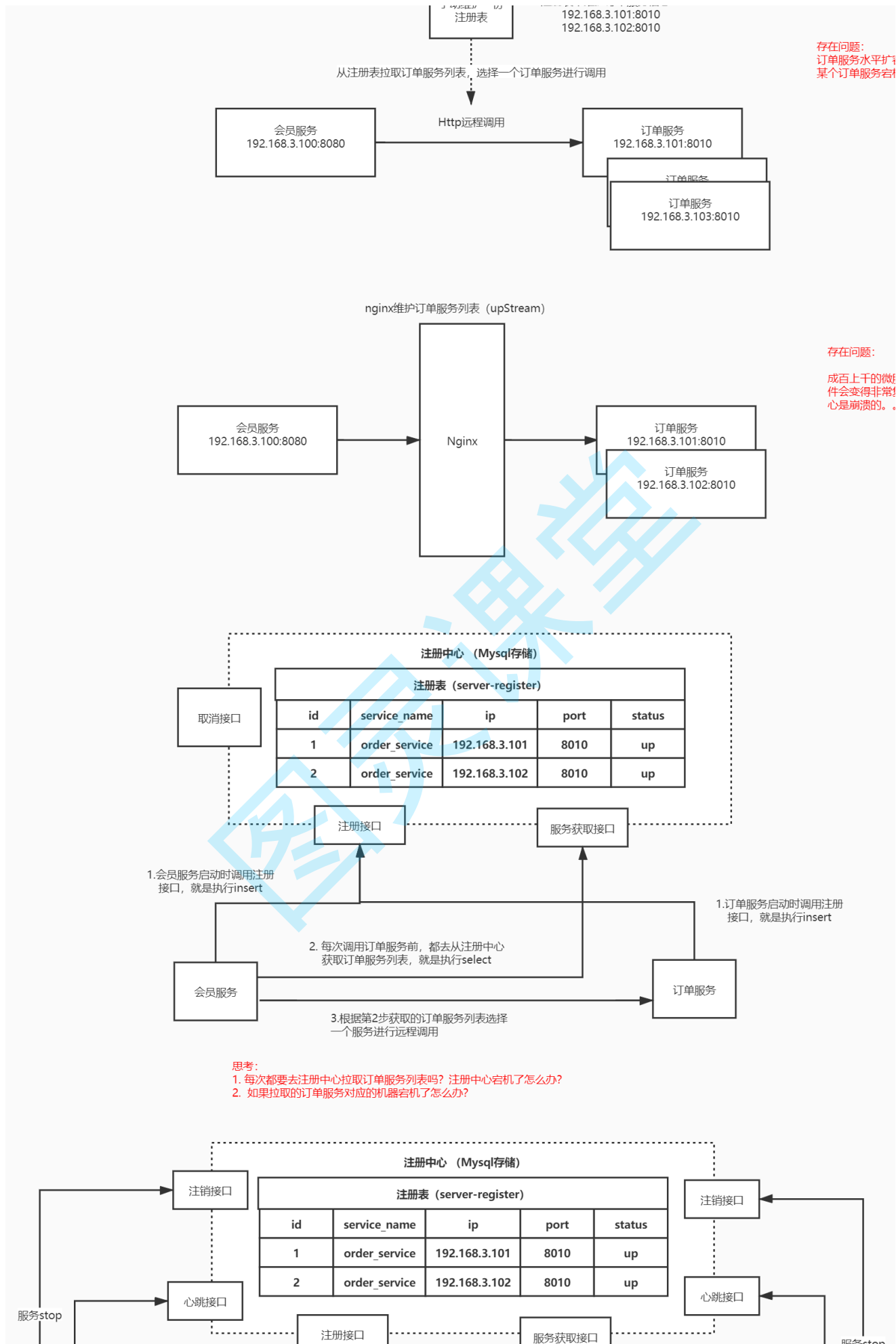


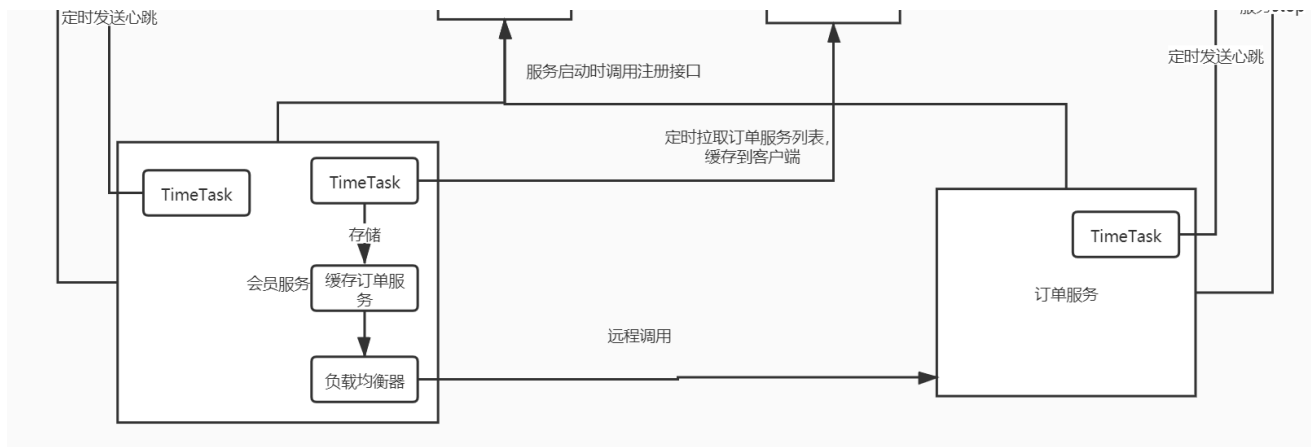
存在问题:

订单服务ip:p
订单服务集群

手动维护一份

注册表中维护订单服务信息:





2.3 核心功能

Nacos Discovery

服务注册: Nacos Client会通过发送REST请求的方式向Nacos Server注册自己的服务, 提供自身的元数据, 比如ip地址、端口等信息。Nacos Server接收到注册请求后, 就会把这些元数据信息存储在一个双层的内存Map中。

服务心跳: 在服务注册后, Nacos Client会维护一个定时心跳来持续通知Nacos Server, 说明服务一直处于可用状态, 防止被剔除。默认5s发送一次心跳。

服务同步: Nacos Server集群之间会互相同步服务实例, 用来保证服务信息的一致性。 leader raft

服务发现: 服务消费者 (Nacos Client) 在调用服务提供者的服务时, 会发送一个REST请求给Nacos Server, 获取上面注册的服务清单, 并且缓存在Nacos Client本地, 同时会在Nacos Client本地开启一个定时任务定时拉取服务端最新的注册表信息更新到本地缓存

服务健康检查: Nacos Server会开启一个定时任务用来检查注册服务实例的健康情况, 对于超过15s没有收到客户端心跳的实例会将其healthy属性置为false(客户端服务发现时不会发现), 如果某个实例超过30秒没有收到心跳, 直接剔除该实例(被剔除的实例如果恢复发送心跳则会重新注册)

主流的注册中心

CAP C 一致性 A可用性 P 分区容错性

	Nacos	Eureka	Consul	CoreDNS	Zookeeper
一致性协议	CP+AP	AP	CP	—	CP
健康检查	TCP/HTTP/MYSQL/Client Beat	Client Beat	TCP/HTTP/gRPC/Cmd	—	Keep Alive
负载均衡策略	权重/metadata/Selector	Ribbon	Fabio	RoundRobin	—
雪崩保护	有	有	无	无	无
自动注销实例	支持	支持	支持	不支持	支持
访问协议	HTTP/DNS	HTTP	HTTP/DNS	DNS	TCP
监听支持	支持	支持	支持	不支持	支持
多数据中心	支持	支持	支持	不支持	不支持
跨注册中心同步	支持	不支持	支持	不支持	不支持
SpringCloud 集成	支持	支持	支持	不支持	支持
Dubbo集成	支持	不支持	支持	不支持	支持
K8S集成	支持	不支持	支持	支持	不支持

雪崩保护:

保护阈值: 设置0-1之间的值 0.6

临时实例: `spring.cloud.nacos.discovery.ephemeral=false`, 当服务宕机了也不会从服务列表中剔除

下图代表永久实例:

临时实例
false

健康实例、不健康实例;

健康实例数/总实例数 < 保护阈值

$1/2 < 0.6$

服务名:

分组:

保护阈值:

元数据:

1

服务路由类型:

ILT

结合负载均衡器 权重的机制, 设置的越大

权重
1
1

3 Nacos Server部署

下载源码编译

源码下载地址: <https://github.com/alibaba/nacos/> 可以用迅雷下载

```
1 cd nacos/  
2 mvn -Prelease-nacos clean install -U  
3 cd nacos/distribution/target/
```

下载安装包

下载地址: <https://github.com/alibaba/Nacos/releases>

3.1 单机模式

官方文档: <https://nacos.io/zh-cn/docs/deployment.html>

解压, 进入nacos目录

```
[root@redis nacos]# ls -l
总用量 32
drwxr-xr-x 4 root root    123 3月  2 21:07 bin
drwxr-xr-x 2 502 games   168 7月 30 14:53 conf
drwxr-xr-x 4 root root    38 3月  2 21:07 data
-rw-r--r-- 1 root root   722 4月 21 15:11 derby.log
-rw-r--r-- 1 502 games 17336 10月 11 2019 LICENSE
drwxr-xr-x 2 root root   4096 4月 21 15:54 logs
-rw-r--r-- 1 502 games 1305 10月 11 2019 NOTICE
drwxr-xr-x 2 root root    30 3月  2 21:01 target
drwxr-xr-x 3 root root    20 4月 21 15:11 work
```

单机启动nacos, 执行命令

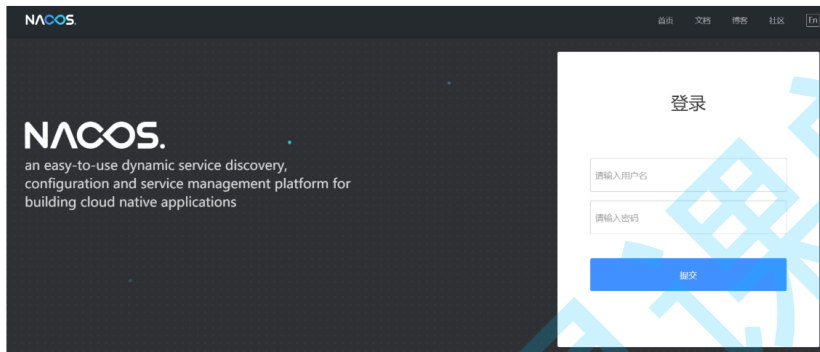
```
1 bin/startup.sh -m standalone
```

也可以修改默认启动方式

```
export SERVER="nacos-server"
export MODE="cluster"
export FUNCTION_MODE="all"
export MEMBER_LIST=""
export EMBEDDED_STORAGE=""
case $opt in
f)
```

可以改为standalone, 单机启动

访问nacos的管理端: <http://192.168.3.14:8848/nacos>, 默认的用户名密码是 nocas/nocas

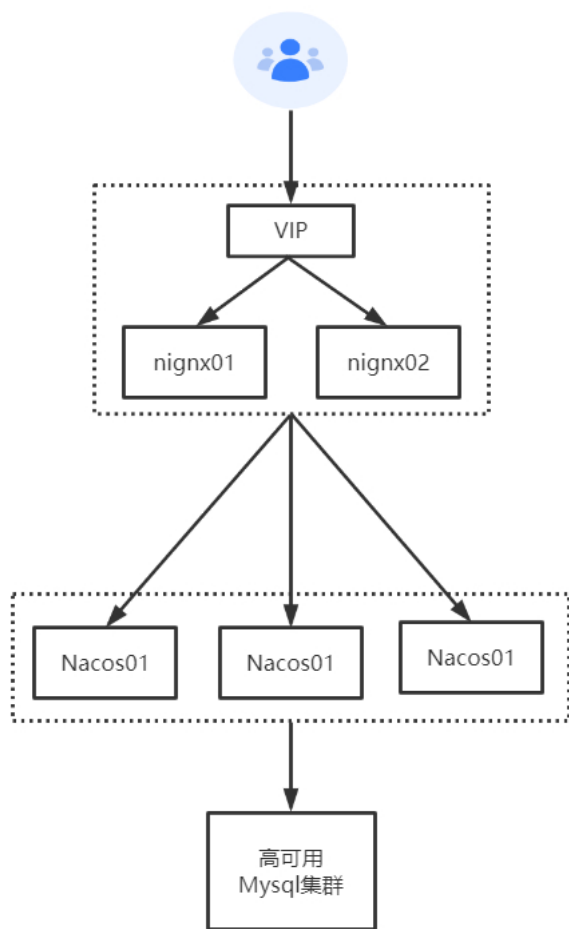


3.2 集群模式

1. jdk1.8+
2. maven 3.3+
3. nginx 作为负载均衡
4. mysql

官网文档: <https://nacos.io/zh-cn/docs/cluster-mode-quick-start.html>

集群部署架构图



1.下载

```
1 mkdir nacos
```

```
1 wget https://github.com/alibaba/nacos/releases/download/1.4.1/nacos-server-1.4.1.tar.gz
```

创建多个nacos server

重复三次

```
1 tar -zxvf nacos-server-1.4.1.tar.gz
```

```
1 mv nacos nacos8849
```

1) 单机搭建伪集群，复制nacos安装包，修改为nacos8849, nacos8850, nacos8851

```
[root@redis nacos-cluster]# ls
nacos8849  nacos8850  nacos8851
```

2) 以nacos8849为例，进入nacos8849目录

2.1) 修改conf/application.properties的配置，使用外置数据源 要使用mysql5.7+ (包括)

```
1 #使用外置mysql数据源
2 spring.datasource.platform=mysql
3
4 ### Count of DB:
5 db.num=1
6
```

```

7  ### Connect URL of DB:
8  db.url.0=jdbc:mysql://127.0.0.1:3306/nacos?characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconn
ect=true&useUnicode=true&useSSL=false&serverTimezone=UTC
9  db.user.0=root
10 db.password.0=root
11

```

```

### If use MySQL as datasource:
spring.datasource.platform=mysql

### Count of DB:
db.num=1

### Connect URL of DB:
db.url.0=jdbc:mysql://127.0.0.1:3306/nacos?characterEncoding=utf8&connectTimeout=1000&
socketTimeout=3000&autoReconnect=true&useUnicode=true&useSSL=false&serverTimezone=UTC
db.user.0=root
db.password.0=root

```

2.2) 将conf\cluster.conf.example改为cluster.conf,添加节点配置

```

1 # ip:port
2 192.168.65.220:8849
3 192.168.65.220:8850
4 192.168.65.220:8851

```

nacos8850, nacos8851 按同样的方式配置。

3) 创建mysql数据库,sql文件位置: conf\nacos-mysql.sql

4) 如果出现内存不足: 修改启动脚本 (bin\startup.sh) 的jvm参数

```

1 JAVA_OPTS="${JAVA_OPTS} -server -Xms512m -Xmx512m -Xmn256 -XX:MetaspaceSize=64m -XX:MaxMetaspaceSize=128m"

```

```

if [[ "${MODE}" == "standalone" ]]; then
    JAVA_OPTS="${JAVA_OPTS} -Xms512m -Xmx512m -Xmn256m"
    JAVA_OPTS="${JAVA_OPTS} -Dnacos.standalone=true"
else
    if [[ "${EMBEDDED_STORAGE}" == "embedded" ]]; then
        JAVA_OPTS="${JAVA_OPTS} -Dnacos.embeddedStorage=true"
    fi
    JAVA_OPTS="${JAVA_OPTS} -server -Xms512m -Xmx512m -Xmn256m -XX:MetaspaceSize=64m -XX:MaxMetaspaceSize=128m"
    JAVA_OPTS="${JAVA_OPTS} -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${BASE_DIR}/
mp.hprof"
    JAVA_OPTS="${JAVA_OPTS} -XX:-UseLargePages"
fi

```

5) 分别启动nacos8849, nacos8850, nacos8851

以nacos8849为例, 进入nacos8849目录, 启动nacos

```

1 bin/startup.sh

```

```

[root@redis nacos8850]# bin/startup.sh
/usr/local/jdk1.8.0_181/bin/java -server -Xms512m -Xmx512m -Xmn256m -XX:MetaspaceSize=128m -XX:Ma
xMetaspaceSize=320m -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPat
h=/usr/local/soft/nacos-cluster/nacos8850/logs/java_heapdump.hprof -XX:-UseLargePages -Djava.ext.d
irs=/usr/local/jdk1.8.0_181/jre/lib/ext:/usr/local/jdk1.8.0_181/lib/ext:/usr/local/soft/nacos-clu
ster/nacos8850/plugins/cmdb:/usr/local/soft/nacos-cluster/nacos8850/plugins/mysql -Xloggc:/usr/loca
l/soft/nacos-cluster/nacos8850/logs/nacos_gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateS
tamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize
=100M -Dnacos.home=/usr/local/soft/nacos-cluster/nacos8850 -Dloader.path=/usr/local/soft/nacos-clu
ster/nacos8850/plugins/health -jar /usr/local/soft/nacos-cluster/nacos8850/target/nacos-server.jar
--spring.config.location=classpath:/,classpath:/config/,file:./,file:./config/,file:/usr/local/s
oft/nacos-cluster/nacos8850/conf/ -logging.config=/usr/local/soft/nacos-cluster/nacos8850/conf/na
cos-logback.xml --server.max-http-header-size=524288
nacos is starting with cluster
nacos is starting, you can check the /usr/local/soft/nacos-cluster/nacos8850/logs/start.out

```

6) 测试

登录 <http://192.168.3.14:8849/nacos> , 用户名和密码都是nacos

NACOS 1.1.4					
public					
节点列表 public					
节点ip	节点状态	集群地址	Leader心跳(ms)	心跳间隔(ms)	
192.168.3.14:8850	FOLLOWER	18	11867	2500	
192.168.3.14:8851	FOLLOWER	0	17359	3000	
192.168.3.14:8849	LEADER	18	17382	2500	

下载nginx

```
1 1. 添加官方源仓库
2 yum install -y yum-utils
3 yum-config-manager --add-repo https://openresty.org/package/centos/openresty.repo
4
5 2. 安装openresty
6 yum install -y openresty
7
8 cd /usr/local/openresty/
```

7) 官方推荐, nginx反向代理

192.168.56.220:8847/nacos/

```
1 upstream nacoscluster {
2     server 127.0.0.1:8849;
3     server 127.0.0.1:8850;
4     server 127.0.0.1:8851;
5 }
6 server {
7     listen 8847;
8     server_name localhost;
9
10    location /nacos/{
11        proxy_pass http://nacoscluster/nacos/;
12    }
13 }
```

```
upstream nacoscluster {
    server 127.0.0.1:8851;
    server 127.0.0.1:8849;
    server 127.0.0.1:8850;
}

server {
    listen 8847;
    server_name localhost;

    #charset koi8-r;

    #access_log logs/host.access.log main;

    location /nacos/ {
        proxy_pass http://nacoscluster/nacos/;
    }
}
```

访问: <http://192.168.3.14:8847/nacos>

1.3 prometheus+grafana监控Nacos (扩展)

<https://nacos.io/zh-cn/docs/monitor-guide.html>

Nacos 0.8.0版本完善了监控系统, 支持通过暴露metrics数据接入第三方监控系统监控Nacos运行状态。

1. nacos暴露metrics数据

```
1 management.endpoints.web.exposure.include=*
```

测试: <http://localhost:8848/nacos/actuator/prometheus>

localhost:8848/nacos/actuator/prometheus

```
# HELP tomcat_global_error_total
# TYPE tomcat_global_error_total counter
tomcat_global_error_total{name="http-nio-8848",} 24.0
# HELP system_cpu_count The number of processors available to the Java virtual machine
# TYPE system_cpu_count gauge
system_cpu_count 12.0
# HELP process_start_time_seconds Start time of the process since unix epoch.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.615290084082E9
# HELP nacos_monitor
# TYPE nacos_monitor gauge
nacos_monitor{module="config",name="longPolling",} 0.0
nacos_monitor{module="config",name="configCount",} 32.0
nacos_monitor{module="naming",name="failedPush",} 0.0
nacos_monitor{module="naming",name="leaderStatus",} 0.0
nacos_monitor{module="config",name="publish",} 0.0
nacos_monitor{module="naming",name="tcpHealthCheck",} 0.0
nacos_monitor{module="config",name="dumpTask",} 0.0
nacos_monitor{module="config",name="notifyTask",} 0.0
nacos_monitor{module="naming",name="totalPush",} 7.0
nacos_monitor{module="naming",name="avgPushCost",} -1.0
nacos_monitor{module="config",name="getConfig",} 6.0
nacos_monitor{module="naming",name="ipCount",} 2.0
nacos_monitor{module="naming",name="mysqlHealthCheck",} 0.0
nacos_monitor{module="naming",name="serviceCount",} 2.0
nacos_monitor{module="naming",name="httpHealthCheck",} 0.0
nacos_monitor{module="naming",name="maxPushCost",} -1.0
# HELP tomcat_global_sent_bytes_total
# TYPE tomcat_global_sent_bytes_total counter
tomcat_global_sent_bytes_total{name="http-nio-8848",} 3.4530029E7
```

2. prometheus采集Nacos metrics数据

启动prometheus服务

```
1 prometheus.exe --config.file=prometheus.yml
```

测试: <http://localhost:9090/graph>

localhost:9090/graph?g0.expr=nacos_monitor&g0.tab=1&g0.stacked=0&g0.range_input=1h

Prometheus Alerts Graph Status Help Classic UI

☐ Enable query history ☐ Use local time ☒ Enable autocomplete

Q nacos_monitor

Table Graph

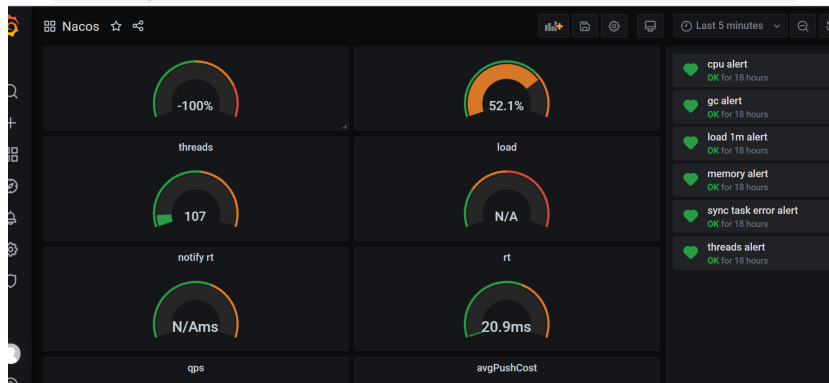
Evaluation time

nacos_monitor{instance="localhost:8848",job="prometheus",module="config",name="configCount"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="config",name="dumpTask"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="config",name="getConfig"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="config",name="longPolling"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="config",name="notifyTask"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="config",name="publish"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="naming",name="avgPushCost"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="naming",name="failedPush"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="naming",name="httpHealthCheck"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="naming",name="ipCount"}
nacos_monitor{instance="localhost:8848",job="prometheus",module="naming",name="leaderStatus"}

3. grafana展示metrics数据

测试: <http://localhost:3000/>

localhost:3000/d/Bz_QALeiz2/nacos?orgId=1



4. Spring Cloud Alibaba Nacos快速开始

4.1 Spring Cloud Alibaba版本选型

组件版本关系

Spring Cloud Alibaba Version	Sentinel Version	Nacos Version	RocketMQ Version	Dubbo Version	Seata Version
2.2.4.RELEASE	1.8.0	1.4.1	4.4.0	2.7.8	1.3.0
2.2.3.RELEASE or 2.1.3.RELEASE or 2.0.3.RELEASE	1.8.0	1.3.3	4.4.0	2.7.8	1.3.0
2.2.1.RELEASE or 2.1.2.RELEASE or 2.0.2.RELEASE	1.7.1	1.2.1	4.4.0	2.7.6	1.2.0
2.2.0.RELEASE	1.7.1	1.1.4	4.4.0	2.7.4.1	1.0.0
2.1.1.RELEASE or 2.0.1.RELEASE or 1.5.1.RELEASE	1.7.0	1.1.4	4.4.0	2.7.3	0.9.0
2.1.0.RELEASE or 2.0.0.RELEASE or 1.5.0.RELEASE	1.6.3	1.1.1	4.4.0	2.7.3	0.7.1

4.2 搭建Nacos-client服务

1) 引入依赖

父Pom中支持spring cloud&spring cloud alibaba, 引入依赖

```
1 <dependencyManagement>
2 <dependencies>
3 <!--引入springcloud的版本-->
4 <dependency>
5 <groupId>org.springframework.cloud</groupId>
6 <artifactId>spring-cloud-dependencies</artifactId>
7 <version>Hoxton.SR3</version>
8 <type>pom</type>
9 <scope>import</scope>
10 </dependency>
11
12 <dependency>
13 <groupId>com.alibaba.cloud</groupId>
14 <artifactId>spring-cloud-alibaba-dependencies</artifactId>
15 <version>2.2.1.RELEASE</version>
16 <type>pom</type>
17 <scope>import</scope>
18 </dependency>
19 </dependencies>
20
21 </dependencyManagement>
```

当前项目pom中引入依赖

```
1 <dependency>
2 <groupId>com.alibaba.cloud</groupId>
3 <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
4 </dependency>
```

2) application.properties中配置

```
1 server.port=8002
2 #微服务名称
3 spring.application.name=service-user
4 #配置 Nacos server 的地址
5 spring.cloud.nacos.discovery.server-addr=localhost:8848
```

更多配置: <https://github.com/alibaba/spring-cloud-alibaba/wiki/Nacos-discovery>

配置项	Key	默认值	说明
服务端地址	spring.cloud.nacos.discovery.server-addr	无	Nacos Server 启动监听的ip地址和端口
服务名	spring.cloud.nacos.discovery.service	\${spring.application.name}	给当前的服务命名
服务分组	spring.cloud.nacos.discovery.group	DEFAULT_GROUP	设置服务所处的分组
权重	spring.cloud.nacos.discovery.weight	1	取值范围 1 到 100，数值越大，权重越大
网卡名	spring.cloud.nacos.discovery.network-interface	无	当IP未配置时，注册的IP为此网卡所对应的IP地址，如果此项也未配置，则默认取第一块网卡的地址
注册的IP地址	spring.cloud.nacos.discovery.ip	无	优先级最高
注册的端口	spring.cloud.nacos.discovery.port	-1	默认情况下不用配置，会自动探测
命名空间	spring.cloud.nacos.discovery.namespace	无	常用场景之一不同环境的注册的区分隔离，例如开发测试环境和生产环境的资源（如配置、服务）隔离等。
AccessKey	spring.cloud.nacos.discovery.access-key	无	当要上阿里云时，阿里云上面的一个云账号名
SecretKey	spring.cloud.nacos.discovery.secret-key	无	当要上阿里云时，阿里云上面的一个云账号密码
Metadata	spring.cloud.nacos.discovery.metadata	无	使用Map格式配置，用户可以根据自己的需要自定义一些和服务相关的元数据信息
日志文件名	spring.cloud.nacos.discovery.log-name	无	
集群	spring.cloud.nacos.discovery.cluster-name	DEFAULT	配置成Nacos集群名称
接入点	spring.cloud.nacos.discovery.endpoint	UTF-8	地域的某个服务的入口域名，通过此域名可以动态地拿到服务端地址
是否集成Ribbon	ribbon.nacos.enabled	true	一般都设置成true即可
是否开启Nacos Watch	spring.cloud.nacos.discovery.watch.enabled	true	可以设置成false来关闭watch

3) 启动springboot应用，nacos管理端界面查看是否成功注册

服务名	分组名称	集群数目	实例数	健康实例数	数据保护策略	操作
service-order	DEFAULT_GROUP	1	1	1	false	详情 删除实例 删除

4) 测试

使用RestTemplate进行服务调用，可以使用微服务名称（spring.application.name）

```
1 String url = "http://service-order/order/findOrderByUserId/"+id;
```

```
2 List<Order> orderList = restTemplate.getForObject(url, List.class);
```

注意: 需要添加@LoadBalanced注解

```
1 @Bean
2 @LoadBalanced
3 public RestTemplate restTemplate() {
4     return new RestTemplate();
5 }
```

4.3 Nacos注册中心架构

