Apusic 金蝶天燕

生成时间：2026-05-19

安装手册

中间件产品 - 金蝶Apusic分布式配置中心V1.0 for zookeeper

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 概述

1.1 摘要

本快速入门指南主要介绍金蝶Apusic分布式配置中心V1.0 for zookeeper（简称“ADCC for zk”）安装、卸载等基本过程，适用于使用金蝶Apusic分布式配置中心 for zookeeper产品进行开发的开发人员、生产环境的系统管理员、运维人员等。

1.2 基础介绍

金蝶Apusic分布式配置中心软件for zookeeper （Apusic Distributed Config Center for zookeeper，简称：ADCC for zk）为分布式应用系统提供高性能协调服务的产品，针对分布式技术而产生的一种新型的配置手段，通过简单的接口能够对分布式应用系统的配置进行统一的管理、实时更新，并支持配置更改的订阅，可实现不需要重启服务器而动态的修改配置文件的内容，有效提高工作效率。。

1.3 术语

ADCC：金蝶Apusic分布式配置中心软件

ADCC_HOME：金蝶Apusic分布式配置中心软件安装目录

2 系统环境要求

系统环境要求如下：

类型	要求
操作系统	国产操作系统如银河麒麟系列、中标麒麟系列、普华、中科红旗、深度等； Windows系列； Linux Red Hat 5.2或以上。
CPU	国产架构：鲲鹏、海光、飞腾、龙芯、兆芯等； 国外架构：如Intel等。
内存、硬盘	建议内存4G或以上；可用空间100G或以上
JDK	JDK8、JDK11、JDK12、JDK17稳定版本

3 安装与卸载

该章节介绍ADCC for zk的安装、卸载操作。

3.1 产品包

安装包通常命名为`ADCC-VXXX-Zookeeper-XXX.tar.gz`，如`ADCC-V1.0.371-Zookeeper-20241219.tar.gz`。

3.2 安装

获取到稳定的ADCC for zk版本后，如`ADCC-V1.0.371-Zookeeper-20241219.tar.gz`；拷贝到目录，如`/opt`；执行解压，得到有目录`adcc/`，此时`/opt/adcc/`就是`${ADCC_HOME}`。

```
tar -zxvf ADCC-V1.0.371-Zookeeper-20241219.tar.gz
```

3.3 卸载

将产品安装目录，如`/opt/adcc`，删除即完成卸载。

注意：卸载前需确认产品已停止运行。

4 启动与停止

该章节介绍ADCC for zk启动、运作等操作。分为单机模式和集群模式。

4.1 Java环境

启动前，先确认已安装有完整的JDK，JDK版本为JDK8及以上。查看java环境命令如下：

```
[user@test3 java]# java -version
java version "1.8.0_231"
Java(TM) SE Runtime Environment (build 1.8.0_231-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.231-b11, mixed mode)
```

4.2 单机模式

4.2.1 启动

进入安装目录`${ADCC_HOME}/bin`，执行启动命令，`adccServer.sh start`。

```
[user@test3 bin]# ./adccServer.sh start
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Starting ADCC ... STARTED
```

4.2.2 停止

进入安装目录`${ADCC_HOME}/bin`，执行停止命令，`adccServer.sh stop`。

```
[user@test3 bin]# ./adccServer.sh stop
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Stopping ADCC ... STOPPED
```

4.2.3 查看状态

进入安装目录`${ADCC_HOME}/bin`，执行查看状态命令，`adccServer.sh status`。结果类似以下内容表示服务已启动，客户端端口为2181，客户端地址为localhost，未启动SSL配置，模式为单机模式。

```
[user@test3 bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: standalone
```

4.3 集群模式

在单机模式下运行ADCC for zk便于评估、某些开发和测试。但在生产中，建议在集群模式下运行ADCC for zk。同一应用程序中的一组复制服务器称为仲裁，在集群模式下，仲裁中的所有服务器都具有相同配置文件的副本。

通常情况下，集群需要保持半数以上的存活节点，因此集群节点至少需要3个及以上。

4.3.1 安装

获取到稳定的ADCC for zk版本后，如`ADCC-V1.0.371-Zookeeper-20241219.tar.gz`；拷贝到目录，如`/opt`；执行解压，得到有目录`adcc/`，此时`/opt/adcc/`就是`${ADCC_HOME}`。

```
tar -zxvf ADCC-V1.0.371-Zookeeper-20241219.tar.gz
```

在集群的每个节点服务器都需要安装ADCC；如果节点在同一服务器，需要注意端口不能冲突。

4.3.2 配置

集群模式的`conf/zoo.cfg`文件与单机模式中使用的类似，但有一些区别。下面以在172.20.140.17、172.20.140.13和172.20.12的机器部署集群为环境进行介绍，在每一个环境上ADCC for zk的`zoo.cfg`的集群配置文件，需要增加如下的最后3行（如果存在端口冲突，则需要修改端口）：

```
tickTime=2000
dataDir=../data
clientPort=2181
initLimit=10
syncLimit=5
server.1=172.20.140.17:2888:3888
server.2=172.20.140.13:4888:5888
server.3=172.20.140.12:6888:7888
```

表单的条目`server.X`列出组成ADCC for zk集群服务的服务器。

```
dataDir=../data
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
#
# The number of snapshots to retain in dataDir
autopurge.snapRetainCount=10
# Purge task interval in hours
# Set to "0" to disable auto purge feature
autopurge.purgeInterval=12

## Metrics Providers
#
# https://prometheus.io Metrics Exporter
#metricsProvider.className=org.apache.zookeeper.metrics.prometheus.PrometheusMetricsProvider
#metricsProvider.httpPort=7000
#metricsProvider.exportJvmInfo=true

#
# the port of adminserver,default is 8080
#admin.serverPort=8080
#
#the config of cluster,must create myid file in dataDir and set value respectively
server.1=172.20.140.17:2888:3888
server.2=172.20.140.13:4888:5888
server.3=172.20.140.12:6888:7888
~
~
```

当服务器启动时，它通过在数据目录中查找文件 `myid` 来知道它是哪个服务器。该文件包含服务器号 `X` ，所以集群配置还需要如下的修改：

在172.20.140.12中的 `dataDir` 目录创建 `myid` 文件，即在 `../data` 下创建 `myid` 文件，并输入上面配置中 `server.1` 中的序号 `1` ，并保存；

同理在172.20.140.17中的 `dataDir` 目录创建 `myid` 文件，即在 `../data` 下创建 `myid` 文件，并输入上面配置中 `server.2` 中的序号 `2` ，并保存；

同理在172.20.140.43中的 `dataDir` 目录创建 `myid` 文件，即在 `../data` 下创建 `myid` 文件，并输入上面配置中 `server.3` 中的序号 `3` ，并保存；

如下图为 `server.1` 服务器的 `myid` 配置：

```
[root@myRabbitA data]# vim myid
[root@myRabbitA data]# ls
myid  version-2
[root@myRabbitA data]# cat myid
1
```

到这里，3个节点的集群就已经配置完成，然后分别启动即可。注意：在3个节点未完全启动完前，会答应连接不上的信息，这是正常的。

上面配置的其他说明：

initLimit：ADCC for zk用来限制quorum中ADCC for zk服务器必须连接到领导者的时间长度的超时。条目syncLimit限制了服务器与Leader之间的过期距离。

对于这两种超时，可以使用tickTime指定时间单位。在本例中，initLimit的超时为5次，每次为2000毫秒，即10秒。

最后，请注意每个服务器名称后面的两个端口号："2888"和"3888"。对等体使用前一个端口连接到其他对等体。这样的连接是必要的，以便对等体可以进行通信，例如商定更新的顺序。更具体地说，ADCC for zk服务器使用此端口将follower连接到Leader。当出现新的Leader时，follower使用此端口打开与Leader的TCP连接。由于默认Leader选举也使用TCP，因此我们当前需要另一个端口进行Leader选举。这是服务器条目中的第二个端口。

4.3.3 验证

依次启动ADCC集群节点。

```
./adccServer.sh start
```

查看ADCC节点状态。

```
./adccServer.sh status
```

下图为示例参考图，主要查看Mode属性值和运行状态，其他值可能会与实际略不同。

可以看到有一个是leader，两个是follower。

```
[root@tck-server bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: leader
[root@tck-server bin]#
```

```
[root@test3 bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[root@test3 bin]#
```

```
[root@myRabbitA bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[root@myRabbitA bin]#
```

先把其中一个follower的节点停止，查看其他节点的状态。可以看到其他节点的状态依旧正常，集群可以正常运行。

```
[root@myRabbitA bin]# ./adccServer.sh stop
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Stopping ADCC ... STOPPED
[root@myRabbitA bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Error contacting service. It is probably not running.
[root@myRabbitA bin]#
```

```
[root@tck-server bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: leader
[root@tck-server bin]#
```

```
[root@test3 bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[root@test3 bin]#
```

再停止一个节点，查看结果。可以看到节点停止后，此时运行的节点少于集群总数量的一半，剩下的一个节点也停止了。

停止运行的节点：

```
[root@tck-server bin]# ./adccServer.sh stop
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Stopping ADCC ... STOPPED
[root@tck-server bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Error contacting service. It is probably not running.
[root@tck-server bin]#
```

没有执行停止运行命令的节点：

```
[root@test3 bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Error contacting service. It is probably not running.
[root@test3 bin]#
```

上一步停止运行的节点：

```
[root@myRabbitA bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Error contacting service. It is probably not running.
[root@myRabbitA bin]#
```

再把其中一个已停止的节点启动，查看结果。会看到集群的节点运行数量过半后，集群又恢复正常运行状态，并且会重新推举出一个leader。

执行启动命令的节点：

```
[root@tck-server bin]# ./adccServer.sh start
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Starting ADCC ... STARTED
[root@tck-server bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[root@tck-server bin]#
```

没有执行停止命令的节点：

```
[root@test3 bin]# ./adccServer.sh status
ADCC JMX enabled by default
Using config: /opt/ADCC/adcc-server-1.0/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: leader
[root@test3 bin]#
```

此时再加入节点也是follower节点，不会影响到现任的leader节点。

5 云部署

ADCC支持云部署。

5.1 build镜像

先获取到ADCC的安装包以及有效的授权文件。

创建Dockerfile。示例如下，操作系统、安装包名称需要根据实际情况修改。

```
FROM harbor.apusic.com/public/jdk1.8:v241
#For ARM Architect
#FROM harbor.apusic.com/arm64/openjdk:8u342-jdk-oraclelinux7

ENV ADCCZK_VERSION 1.0
COPY adcc-server-1.0.tar.gz /home/apusic/

RUN mkdir -p /home/apusic/adcc/data /home/apusic/adcc/wal
/home/apusic/adcc/log
RUN cd /home/apusic &&  \
tar xf adcc-server-1.0.tar.gz -C /home/apusic && \
rm -rf adcc-server-1.0.tar.gz

ADD license.xml /home/apusic/adcc/license.xml
ADD zoo.cfg /home/apusic/adcc/conf/zoo.cfg
COPY entrypoint.sh /home/apusic/adcc/bin/

ENV PATH=/home/apusic/adcc/bin:${PATH} \
    ZOO_LOG_DIR=/home/apusic/adcc/log \
    ZOO_LOG4J_PROP="INFO, CONSOLE, ROLLINGFILE" \
    JMXPORT=9010

RUN chmod +x /home/apusic/adcc/bin/*.sh
ENTRYPOINT ["entrypoint.sh"]
CMD [ "/home/apusic/adcc/bin/adccServer.sh", "start-foreground" ]
EXPOSE 2181 2888 3888 9010
```

执行`docker build`命令，如

```
docker build -t apusic/adcc-zk:v1.0 .
```

5.1.1 验证镜像

使用`docker images`验证镜像。

```
docker images
```

5.1.2 运行容器

执行`docker run`命令运行容器，如：

```
docker run --name adcczk -p 2181:2181 -d apusic/adcc-zk:v1.0
```

执行`docker ps -a` 命令查看容器。

6 产品授权

ADCC需要有对应的许可证才能正常使用，通常情况下，金蝶天燕会根据用户购买的产品版本配套对应的许可证。

产品授权方式分为普通授权和集中授权。

6.1 普通授权

普通授权指根据IP、域名等方式生成 `license.xml` 文件，将授权文件放置安装目录下，`${ADCC_HOME}/license.xml` 。

6.2 集中授权

集中授权指连接授权中心，进行统一授权。需要先搭建金蝶Apusic授权中心，操作方式可参考《金蝶Apusic许可授权中心用户手册》，或联系金蝶天燕技术支持人员。

在服务器配置环境变量，或在ADCC安装目录根目录创建 `acls.properties` 文件，添加以下参数：

```
apusic_acls_enable=true
apusic_acls_authUrls=172.24.4.166:6886
apusic_acls_ns=apusic
apusic_acls_tenant=ApusicTest
```

连接参数说明：

参数名	参数值说明
apusic_acls_enable	是否开启授权中心认证，取值为true或false，为true则表示开启授权中心认证。没有该参数或该参数值为false，都表示没有开启授权中心认证；
apusic_acls_authUrls	授权中心的地址，可设置多个授权地址，格式为ip1:port1,ip2:port2，如果一个授权地址链接失败，会轮询其他的地址；如果开启授权中心认证，则为必填参数，其中端口为授权中心的https端口；
apusic_acls_ns	设置该实例所属的命名空间名称，可选参数；默认值为public，具体的命名空间可以在授权中心管理控制台-系统管理-授权管理查看。
apusic_acls_tenant	设置该实例所属的租户名称，可选参数。

ADCC启动时将会自动连接到Apusic授权中心。

6.3 获取特征码

如果在使用过程中出现许可证过期或无效等问题，建议优先联系对接的天燕服务人员，重新申请对应许可证。重新申请对应许可证时，需要将产品的特征码(auth code)提供到天燕对接人员。

在`${ADCC_HOME}/bin`，执行`adccServer.sh -ac host`，host取值为ip地址或者网卡名称，类似如下:

```
./adccServer.sh -ac 172.20.140.17
```

打印特征码信息，类似如下，Auth Code=右边则为特征码内容：

```
ADCC JMX enabled by default
Auth Code=SZTY777387783
```

获取特征码后再提供特征码申请授权文件。

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com