**docker环境运维部署手册**

目录

[一、 面向对象 3](#_Toc100735500)

[二、 系统功能 3](#_Toc100735501)

[三、 前期准备工作 3](#_Toc100735502)

[3.1安装docker 3](#_Toc100735503)

[3.2加载docker镜像文件 3](#_Toc100735504)

[四、 docker镜像部署 4](#_Toc100735505)

[4.1 在线asr docker部署 4](#_Toc100735506)

[4.1.1 上传文件 4](#_Toc100735507)

[4.1.2 启动镜像 4](#_Toc100735508)

[4.1.3启动服务,配置相关参数 4](#_Toc100735509)

[4.1.4验证服务 7](#_Toc100735510)

[4.2 离线转写 docker部署 8](#_Toc100735511)

[4.2.1 上传文件 8](#_Toc100735512)

[4.2.2 启动镜像 8](#_Toc100735513)

[4.2.3 启动服务 配置相关参数 8](#_Toc100735514)

[4.2.4验证服务 10](#_Toc100735515)

[五、 工具包更新 11](#_Toc100735516)

[5.1 redis更新 11](#_Toc100735517)

[5.1.1在线引擎: 11](#_Toc100735518)

[5.1.2 离线引擎 11](#_Toc100735519)

[5.2 jdk更新 11](#_Toc100735520)

[5.2.1 在线引擎 11](#_Toc100735521)

[5.2.2 离线引擎 12](#_Toc100735522)

[5.3 tomcat更新 12](#_Toc100735523)

[5.3.1 在线引擎 12](#_Toc100735524)

[5.3.2 离线引擎 12](#_Toc100735525)

[六、 服务维护操作 16](#_Toc100735526)

[6.1在线引擎 16](#_Toc100735527)

[6.1.1 日常巡检操作 16](#_Toc100735528)

[6.1.2 验证服务是否正常工作 17](#_Toc100735529)

[6.1.3查看服务日志 17](#_Toc100735530)

[6.2离线引擎 17](#_Toc100735531)

[6.2.1 日常巡检操作 17](#_Toc100735532)

[6.2.2 验证服务是否正常工作 18](#_Toc100735533)

[6.2.3 查看服务日志 19](#_Toc100735534)

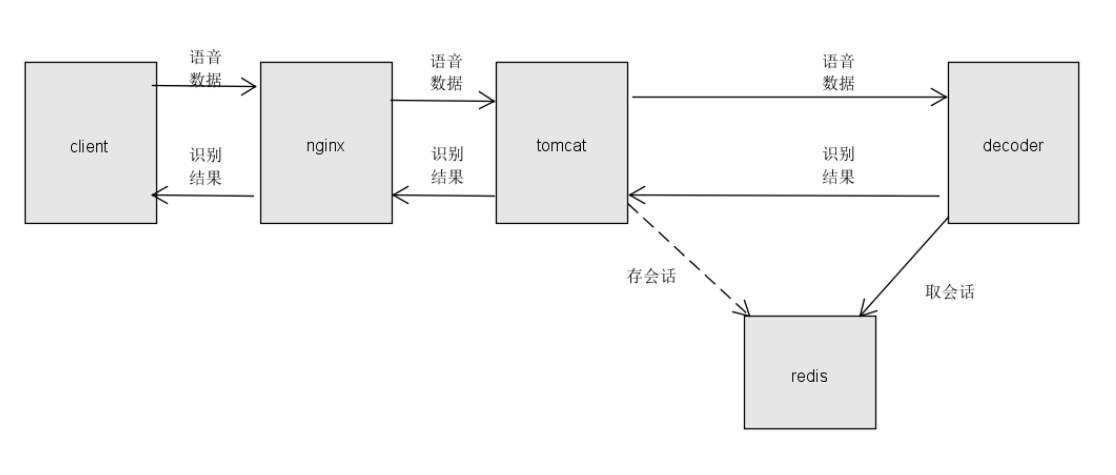
# 面向对象

该文档面向是运维人员，提供部署上的技术支持。在进行部署时,参考此文档进行操作,确保我们的服务正常运行

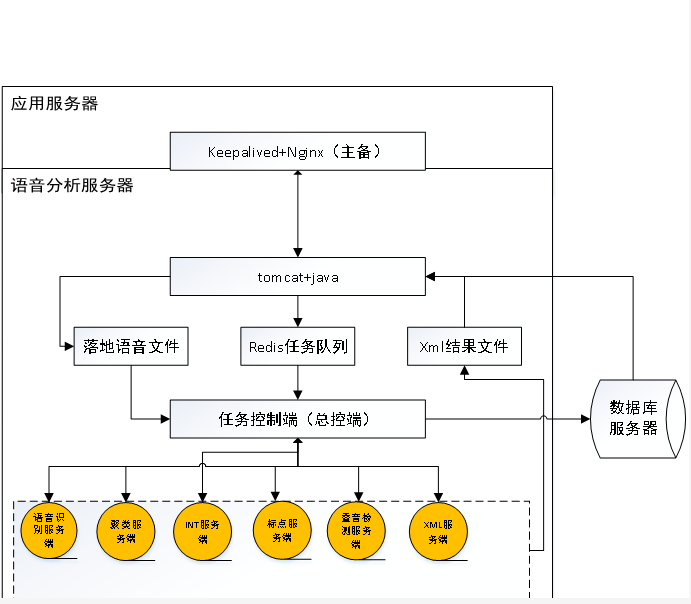
# 系统功能

|  |  |
| --- | --- |
| 系统服务 | 系统功能 |
| Asr-e2e | 在线导航语音识别 |
| 离线-e2e | 离线语音转写 |

## 2.1 在线引擎模块结构图



## 2.2 离线引擎模块结构图



# 前期准备工作

## 3.1安装docker

在提供的部署安装包中有提供docker的离线安装包docker.tar.gz，将其放入到服务机中

解压: **tar -zxvf docker.tar.gz -C docker**

进入到解压目录: **cd docker**

使用rpm安装: **rpm -Uvh --force --nodeps \*.rpm**

## 3.2加载docker镜像文件

系统分为在线导航识别和在线辅助以及离线语音转写，因此加载的docker镜像文件有两个，分别是在线和离线的 offline\_base.tar(离线)、online\_docker.tar(在线)

如果是部署在线导航识别、在线辅助，则使用online\_docker.tar

加载方式: **docker load -i online\_docker.tar**

如果使用离线转写系统，则使用offline\_docker.tar

加载方式: **docker load -i offline\_docker.tar**

加载完成后，使用**docker images** 查看加载镜像记录，如下所示；



# docker镜像部署

## 4.1 在线asr docker部署

### 4.1.1 上传文件

在线服务docker环境为挂载方式部署，包含部署包和镜像两个文件，在部署docker的物理机上使用当前用户创建online目录，online目录的绝对路径为 /home/当前用户名/online，将部署包通过上传工具放到online目录下。

### 4.1.2 启动镜像

docker run -p hostPort:containerPort -i -t -d -v /home/当前用户名/online:/root --name online(容器名字) online:v1.1 /bin/bash

**参数介绍：**

-p： 端口映射，hostPort物理机端口，containerPort容器端口；

-v： 目录映射，物理机目录：容器目录；

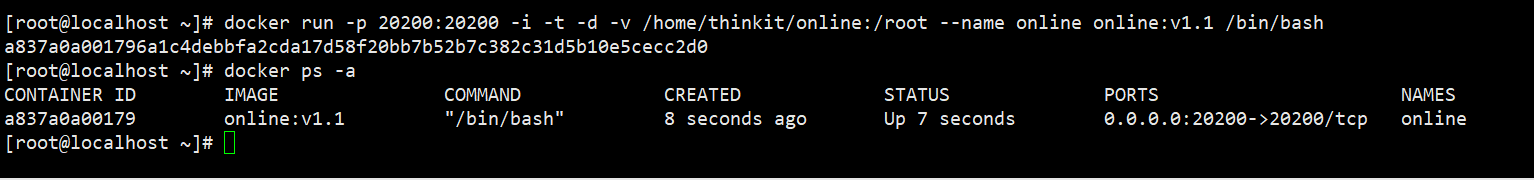
-d：后台运行容器；

-it：以交互模式启动容器；

--name：容器名字

**启动镜像: docker run -p 20200:20200 -i -t -d -v /home/当前用户名/online:/root --name online online:v1.1 /bin/bash**

**查看镜像状态: docker ps -a**



### 4.1.3启动服务,配置相关参数

**进入容器:** docker exec -it <CONTAINER ID> bash

**docker exec -it a837a0a00179 bash**

<CONTAINER ID>: 通过docker ps -a 可以查看到该值

容器内操作插件包括：安装jdk、redis、tomcat和Decoder识别服务

**进入/root目录: cd /root**

**解压部署包: tar -zxvf Rel\_OnlineASR-e2e\_XXX.tar.gz**

**进入安装目录:cd rel\_onlineasr/tools/tools\_install**

**安装jdk: sh 01\_install\_jdk.sh && source ~/.bashrc**

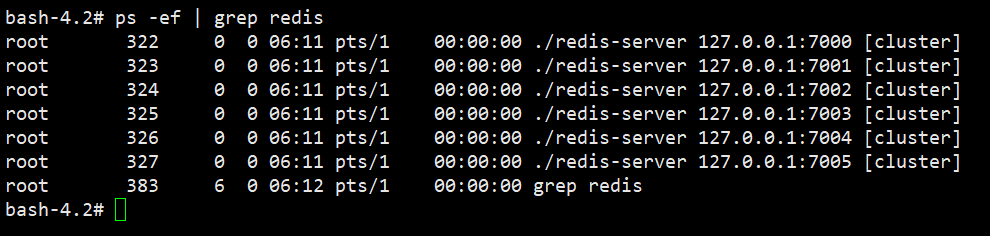
**安装redis: sh 02\_install\_redis.sh**

**安装tomcat: sh 03\_install\_tomcat.sh**

**启动redis: cd /root/local/redis/redis-3.2.0 && sh cluster\_install\_ms.sh**

**查看redis启动状态:**

**ps -ef | grep redis**

****

**配置启动tomcat:**

**cd /root/local/tomcat/apache-tomcat-9.0.35/webapps/asrability/WEB-INF/classes**

**vi redis.properties**

修改

****

进入到/root/local/tomcat/apache-tomcat-9.0.35/bin 目录

**cd /root/local/tomcat/apache-tomcat-9.0.35/bin**

关闭tomcat:**sh shutdown.sh**

启动tomcat:**sh start.sh**

**配置decoder**

**cd /root/rel\_onlineasr/Decoder/conf**

**vi decoder.conf**



注意**/root/local/tomcat/apache-tomcat-9.0.35/webapps/asrability/WEB-INF/classes**

**/redis.properties中的 ASR\_REDISSETLIST = ASR\_SERVICEREQ:test01要和**

**/root/rel\_onlineasr/Decoder/conf中的 RedisKey=ASR\_SERVICEREQ:test01 相同**

**申请授权 启动decoder**

**cd /root/rel\_onlineasr/tools**

执行**./getinfo**

生成\*\_machine.info结尾的信息文件，取出改信息文件，发给我方，进行授权申请;

进入到目录: **cd /root/rel\_onlineasr/Decoder/bin**

将授权申请后的文件放在目录 **/root/rel\_onlineasr/Decoder/bin**

修改授权文件名字为: **license\_120.dat**

验证服务能否正常启动: **sh decoder.sh**

如果服务能正常启动: **sh start.sh** 启动服务;

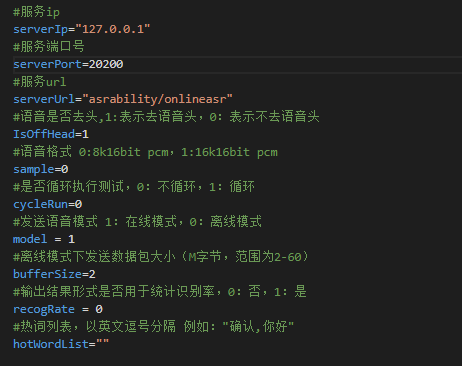
### 4.1.4验证服务

进入目录 **cd /root/rel\_onlineasr/tools/SDK/Linux\_Cpp**

解压: **tar -zxvf C++\_demo.tar.gz**

**cd C++\_demo**

修改config.ini文件  **vi config.ini**



执行 **sh run.sh;**有识别结果输出，表示服务启动成功。

**退出容器**

exit 即可退出容器

## 4.2 离线转写 docker部署

### 4.2.1 上传文件

离线服务docker环境为挂载方式部署，包含部署包和镜像两个文件，在部署docker的物理机上使用当前用户创建offline目录，offline目录的绝对路径为 /home/当前用户名/offline，将部署包Thinkit\_Offline\_SystemE-XXX.tar.gz解压到offline目录下。

### 4.2.2 启动镜像

**加载镜像：**

**docker load -i offline\_base.tar**

**docker images** 查看加载镜像记录，如下所示；



**启动镜像**

docker run -p hostPort:containerPort -i -t -d -v /home/当前用户名/tempvoice:/root/tempvoice -v /home/当前用户名/tempvoice\_xml:/root/tempvoice\_Xml -v /home/当前用户名/offline/thinkit\_offline\_systeme:/root/thinkit\_offline\_systeme --name offline(容器名字) offline:v1.1 --privileged=true offline\_base:v1.1 /usr/sbin/init

**docker run -p 20100:20100 -i -t -d -v /home/当前用户名/tempvoice:/root/tempvoice -v /home/当前用户名/tempvoice\_xml:/root/tempvoice\_Xml -v /home/当前用户名/offline/thinkit\_offline\_systeme:/root/thinkit\_offline\_systeme --name offline(容器名字) offline:v1.1 --privileged=true offline\_base:v1.1 /usr/sbin/init**

**参数介绍：**

-p： 端口映射，hostPort物理机端口，containerPort容器端口；

-v： 目录映射，物理机目录：容器目录；

-d：后台运行容器；

-it：以交互模式启动容器；

--name：容器名字

--privileged:容器内的root用外部root权限；

### 4.2.3 启动服务 配置相关参数

**进入容器:** docker exec -it <CONTAINER ID> bash

**docker exec -it a837a0a00179 bash**

<CONTAINER ID>: 通过docker ps -a 可以查看到该值

容器内操作插件包括：安装jdk、redis、tomcat和Decoder识别服务

**进入/root目录，**

cd /root

进入目录thinkit\_offline\_systeme/Thinkit\_Offline\_SystemE

**cd thinkit\_offline\_systeme/Thinkit\_Offline\_SystemE**

**修改mysql密码**

cd database

vi db\_user\_pw 修改PASSWORD 为123456

**申请授权**

**cd /root/thinkit\_offline\_systeme**

1.需要申请的license如下:

Thinkit\_Offline\_SystemE/System\_Offline\_Server/ClusterSceneServer/license.dat##产品号26

Thinkit\_Offline\_SystemE/System\_Offline\_Server/DetectOverlap/license.dat##产品号199

Thinkit\_Offline\_SystemE/System\_Offline\_Server/TBNR\_release\_time/bin/license\_120.dat##产品号120（需替换4.3版本的授权文件）

/Thinkit\_Offline\_SystemE/System\_Offline\_Server/PuncRecognize//license.dat##产品号120

2.申请流程

执行Thinkit\_Offline\_SystemE/System\_Offline\_Server/tools/getinfo脚本，当前目录得到机器信息文件localhost.localdomain\_machine.info。

执行Thinkit\_Offline\_SystemE/System\_Offline\_Server/TBNR\_release\_time/bin/getinfo\_4.3脚本，当前目录得到机器信息文件localhost.localdomain\_machine.info。(识别模块授权使用4.3版本，只需申请120号授权)

2.3 发送localhost.localdomain\_machine.info和getinfo\_4.3生成的机器信息（localhost.localdomain\_machine.info）文件给北京相关人员，标明秘钥产品号和服务器线数（一般为CPU核心数的1/2），申请秘钥（需申请两个版本的120号授权）。

2.4替换对应的产品秘钥。

**启动服务**

**./SystemStart.sh**

查看识别服务器检查System\_Offline\_Server目录下是否有RunAndMonitor.log文件生成？

如果有RunAndMonitor.log生成，且RunAndMonitor.log文件末尾有类似如下内容输出（重点看是否有custom\_client\_multiserver is running），则表示引擎启动成功。

Sat Sep 29 04:00:24 EDT 2018

0 offline\_customer\_server\_test\_dnnvad running,will restart soon !

restart all server and client...

FilesSync\_Server startup OK !

Wait for client initial...

offline\_customer\_server\_test\_dnnvad is running

Cluster\_Scene\_server\_test is running

itn\_recognize\_server is running

punc\_recognize\_server is running

DetectOverlapPro is running

DetectEmotionPro is running

gender\_recognize\_server is running

XMLServer is running

Wait for client initial...

custom\_client\_multiserver is running

Sat Sep 29 04:01:40 EDT 2018

*注意：执行startup.sh启动各引擎服务，到引擎初始化完毕整个过程约耗时3分钟。*

### 4.2.4验证服务

进入到目录

**/root/thinkit\_offline\_systeme/Thinkit\_Offline\_SystemE/ System\_Offline\_Server/tools/SDK/ Linux\_Cpp**,

解压demo\_3.1.tar.gz文件，进入目录demo，

修改文件test.sh

* ThreadNum=1
* voicedir=./test
* Url=http://127.0.0.1:20100/ability/offlineasr

设置环境变量 export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:./

然后执行 sh test.sh

在/root/tempvoice\_Xml目录中生成xml文件，表示验证成功。

**退出容器**

执行exit，退出当前容器。

# 工具包更新

注意: 所有的更新操作 均是在容器内进行修改

## 5.1 redis更新

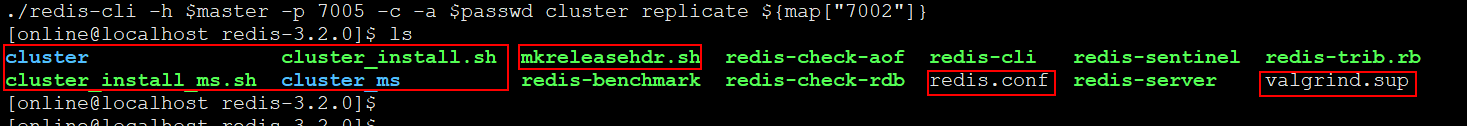
### 5.1.1在线引擎:

1. 从官网下载需要更新的redis版本,编译安装,将编译好的包放置到~/local/下

2. 进入到~/local/目录下,将相关的脚本和配置文件拷贝到新版本目录下

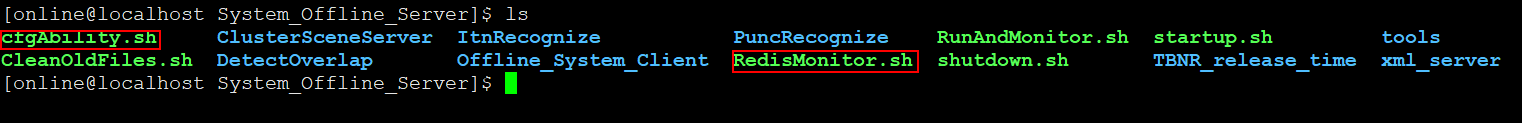
3. 根据采用的redis模式(单机、集群) 启动redis-server即可

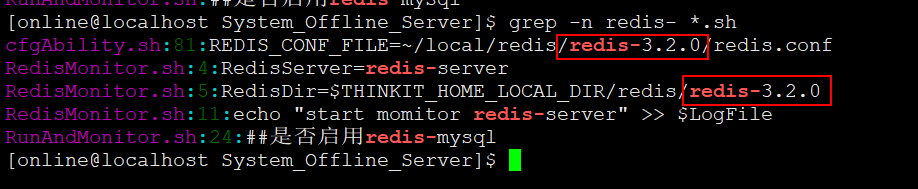
涉及到的配置文件和脚本:



### 5.1.2 离线引擎

在上述的基础上改动几个脚本中的值 分别如下:





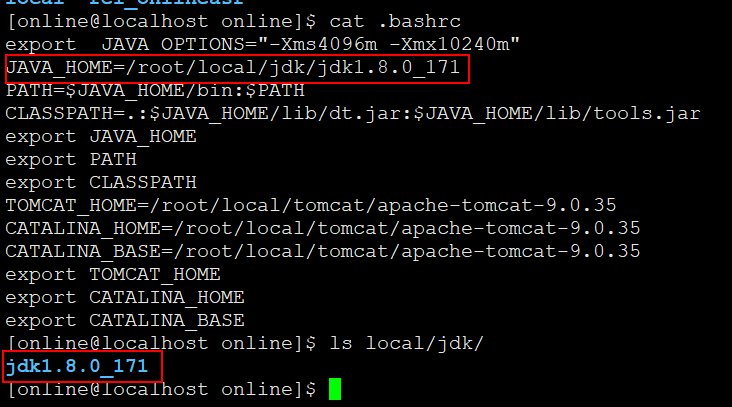
改成对于的安装目录 假设新的redis安装在redis-4.2.0 则将redis-3.2.0 改成redis-4.2.0即可

## 5.2 jdk更新

### 5.2.1 在线引擎

1、官网上下载需要更新的jkd版本，解压到~/local/jdk即可完成版本更新。

2. 环境变量(.bashrc)修改： 修改JAVA\_HOME的值



### 5.2.2 离线引擎

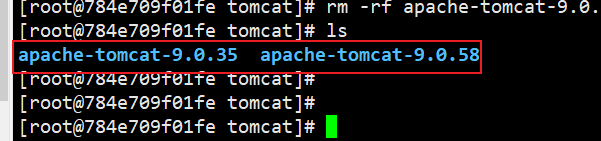
同上述方式 都是在~/local目录下

## 5.3 tomcat更新

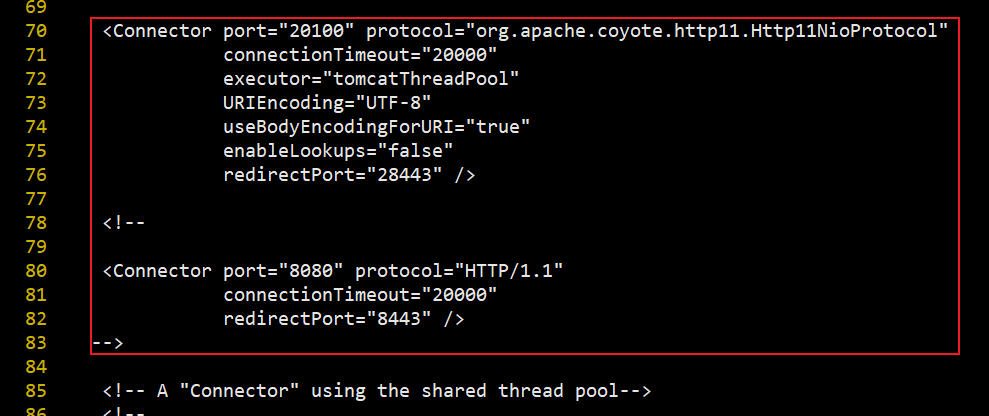
### 5.3.1 在线引擎

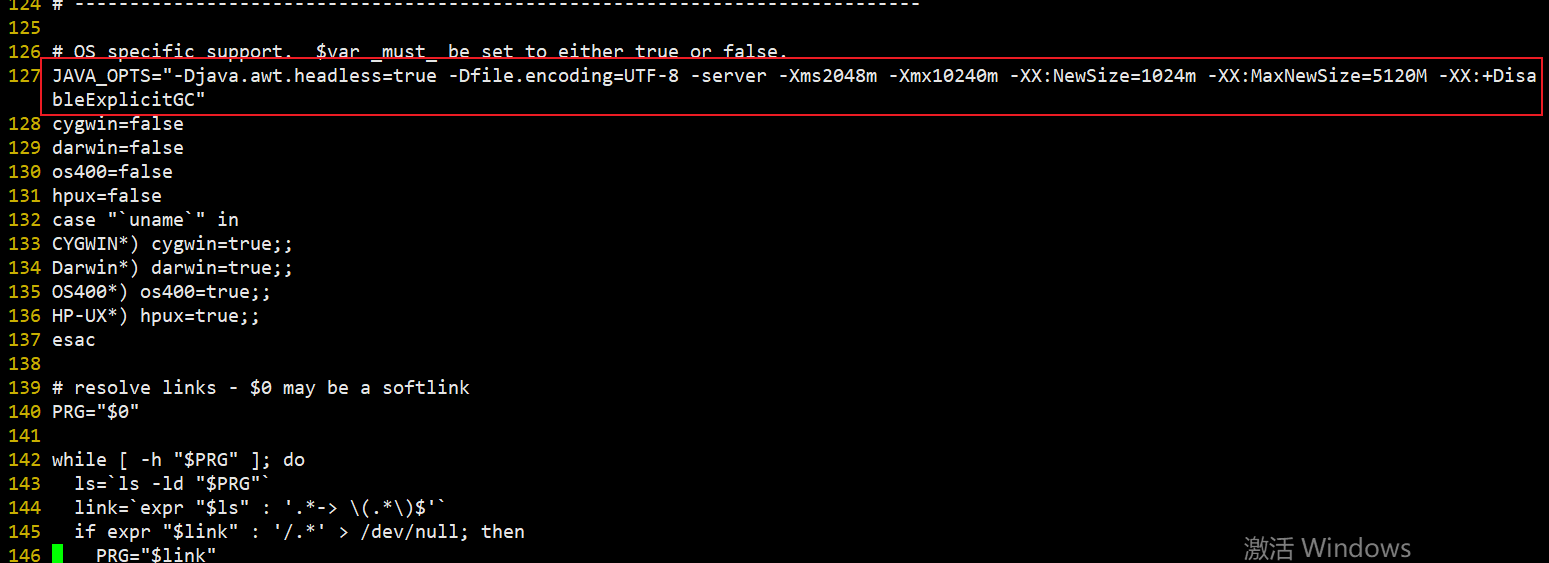
去掉离线引擎中的第4步即可

### 5.3.2 离线引擎

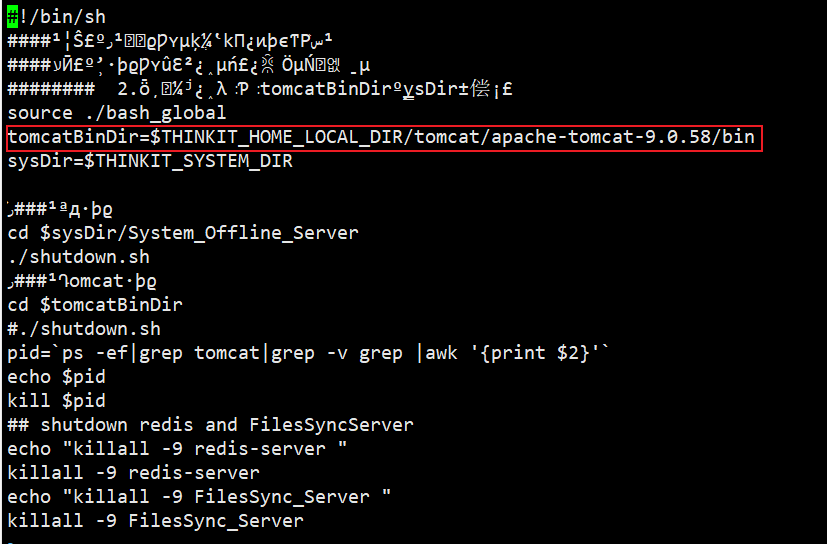
假设从apache-tomcat-9.0.53 升级到 apache-tomcat-9.0.58  


1. 从tomcat官网(https://tomcat.apache.org/)下载要升级的安装包
2. 将下载下来的tomcat包解压后，删除webapps目录的文件，同时修改server.xml 和catalina.sh文件

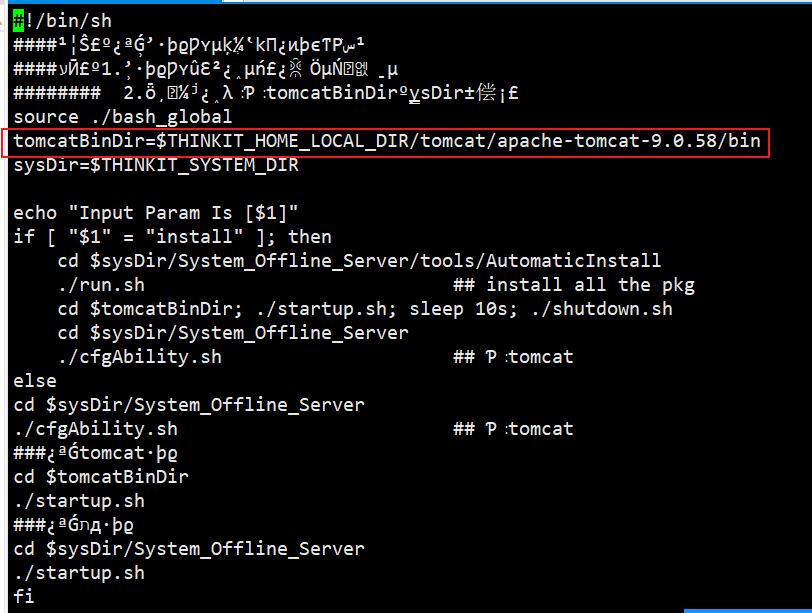
server.xml  


catalina.sh  


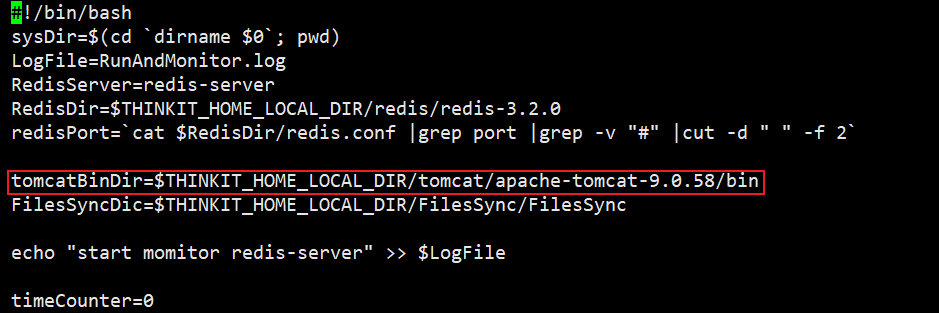
1. 将旧版本tomcat中的webapps目录下的文件全部拷贝到新版本的tomcat的webapps目录下
2. 修改SystemShutdown.sh 、SystemStartup.sh、RedisMonitor.sh、cfgAbility.sh

SystemShutdown.sh  


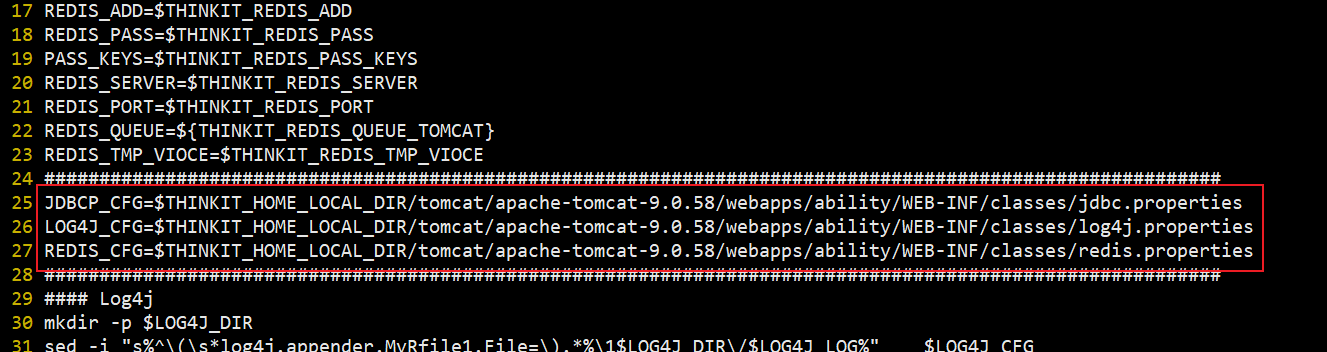
SystemStartup.sh

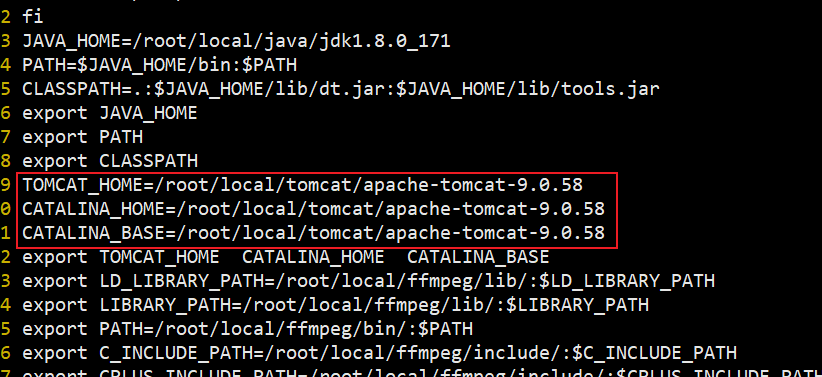


RedisMonitor.sh

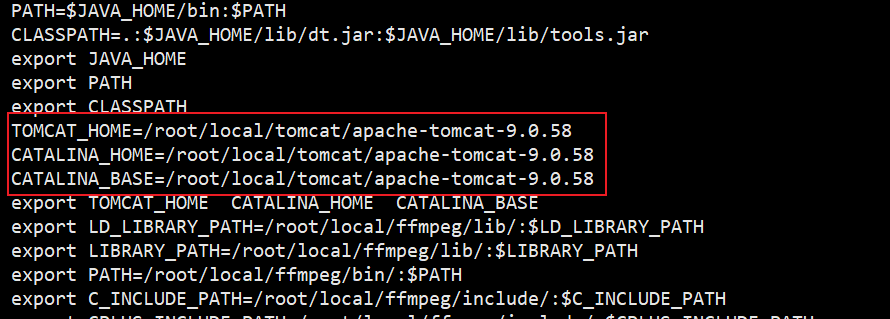


cfgAbility.sh



1. 修改~/.bashrc、~/.bash\_profile,然后执行 source ~/.bashrc、source ~/.bash\_profile  
   ~/.bashrc  
   

~/.bash\_profile



1. 重启离线引擎服务

# 服务维护操作

## 6.1在线引擎

### 6.1.1 日常巡检操作

在线识别系统有四个工作组件,tomcat、redis和decoder；

进入到容器中: docker exec -it online bash

根据服务部署列表，对每个插件进行日常巡检；具体操作如下：

检测各个进程是否存在：

执行ps -ef | grep tomcat， 查看进程是否正常工作，出现类似于下图的结果



执行ps -ef | grep redis， 查看进程是否正常工作，出现类似于下图的结果



执行ps -ef | grep decoder， 查看进程是否正常工作，出现类似于下图的结果



### 6.1.2 验证服务是否正常工作

退出容器，进入如下目录:

/home/thinkit/online/rel\_onlineasr/tools/SDK/Linux\_Cpp/C++\_demo

执行 sh run.sh 如果有如下结果输出，则表示服务正常。如果出现问题，可通过查看下方的服务日志定位错误。

### 6.1.3查看服务日志

服务日志路径如下：

tomcat:/root/local/tomcat/apache-tomcat-8.5.34/log

decoder: /root/rel\_onlineasr/Decoder/log

当服务中某个会话出现异常时，可根据日志中的会话id信息，分析当前会话异常的具体原因。

## 6.2离线引擎

### 6.2.1 日常巡检操作

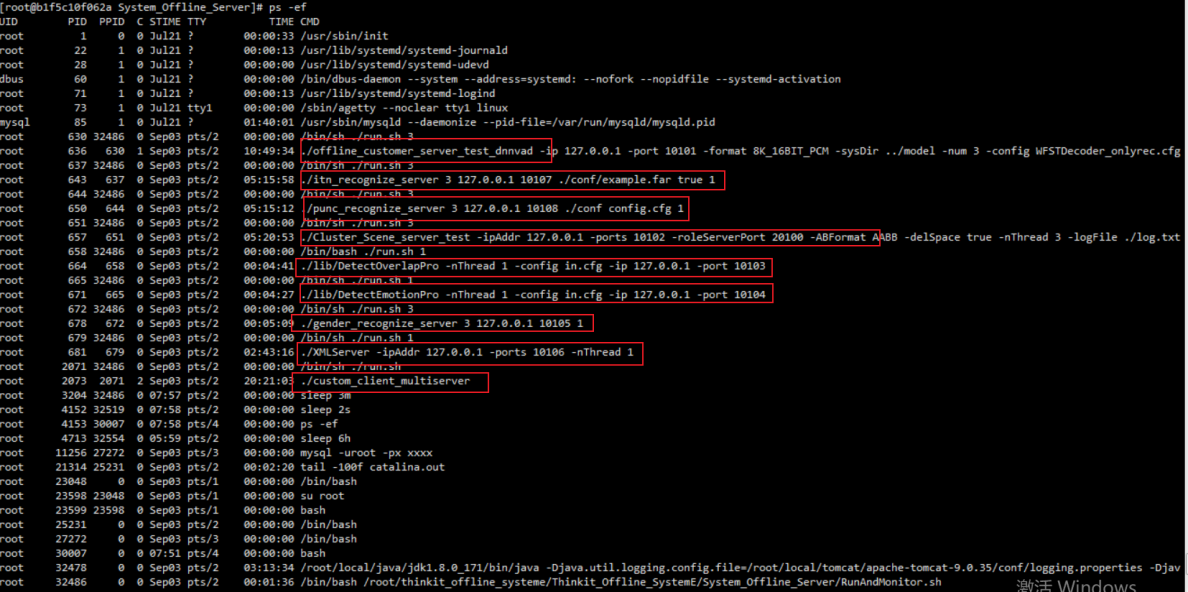
进入到容器 docker exec -it offfline bash

1. 查看组件是否正常工作

离线引擎系统工作组件有：Redis、MySQL、离线引擎，其中Redis、MySQL是由运维提供的。离线引擎的工作组件有：Tomcat、offline\_customer\_server\_test\_dnnvad、itn\_recognize\_server、Cluster\_Scene\_server、DetectOverlapPro、gender\_recognize\_server、XMLServer、

custom\_client\_multiserver。

离线系统是否正常运行可以通过进程是否正常存在来检测，命令如下：



正常运行的情况下就可以看到进程Tomcat、offline\_customer\_server\_test\_dnnvad、itn\_recognize\_server、Cluster\_Scene\_server、DetectOverlapPro、DetectEmotionPro、

gender\_recognize\_server、XMLServer、

custom\_client\_multiserver等都在正常运行。

1. 分析MYSQL记录

1) 登陆mysql： mysql -uroot -p123456

2）查看task\_info表中的信息

查询voiceStatus=1的语音条数,一般每台服务器24条左右；而且creatTime应该是最近几分钟

查询voiceStatus=3的语音 failCause：

a), voice not exits 待处理语音不存在

b), can't open xmlfilepath xml文件路径配置错误

c), voice transcode failed 语音转码失败

d), voice is too short 语音过短，小于1s

e), Can not handle the abnormal audio! 语音连续两次转写失败（严重错误）

f), open xmlfile failed xml文件不存在

g), task failed where server= \*\* 引擎的\*\*模块处理失败、

### 验证服务是否正常工作

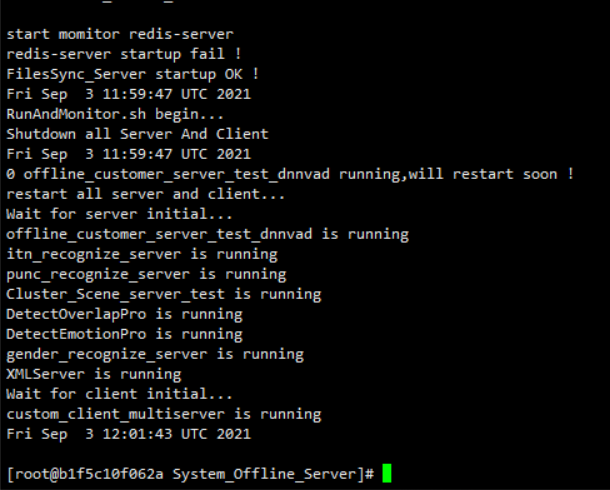
1. 通过日志查看服务是否正常启动

进入如下目录:

/root/thinkit\_offline\_systeme/Thinkit\_Offline\_SystemE/

System\_Offline\_Server/ 查看RunAndMonitor.log 文件

看是否有如下结果出现:



1. 检查服务是否能够正常使用

进入到如下目录:

/root/thinkit\_offline\_systeme/Thinkit\_Offline\_SystemE/

System\_Offline\_Server/tools/SDK/ Linux\_Cpp/C++\_demo

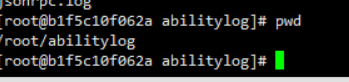
执行export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:./

执行sh test.sh 在/root/tempvoice\_Xml目录中生成xml文件，表示服务可以正常成功。

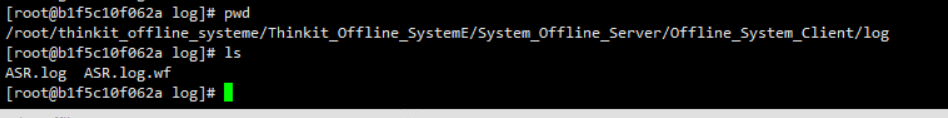
### 6.2.3 查看服务日志

服务日志路径如下：

tomcat日志：jsonrpc.log



离线系统日志: ASR.log、ASR.log.wf、



当服务中离线任务处理异常时，可根据日志中的错误信息，分析离线语音处理异常的详细原因。