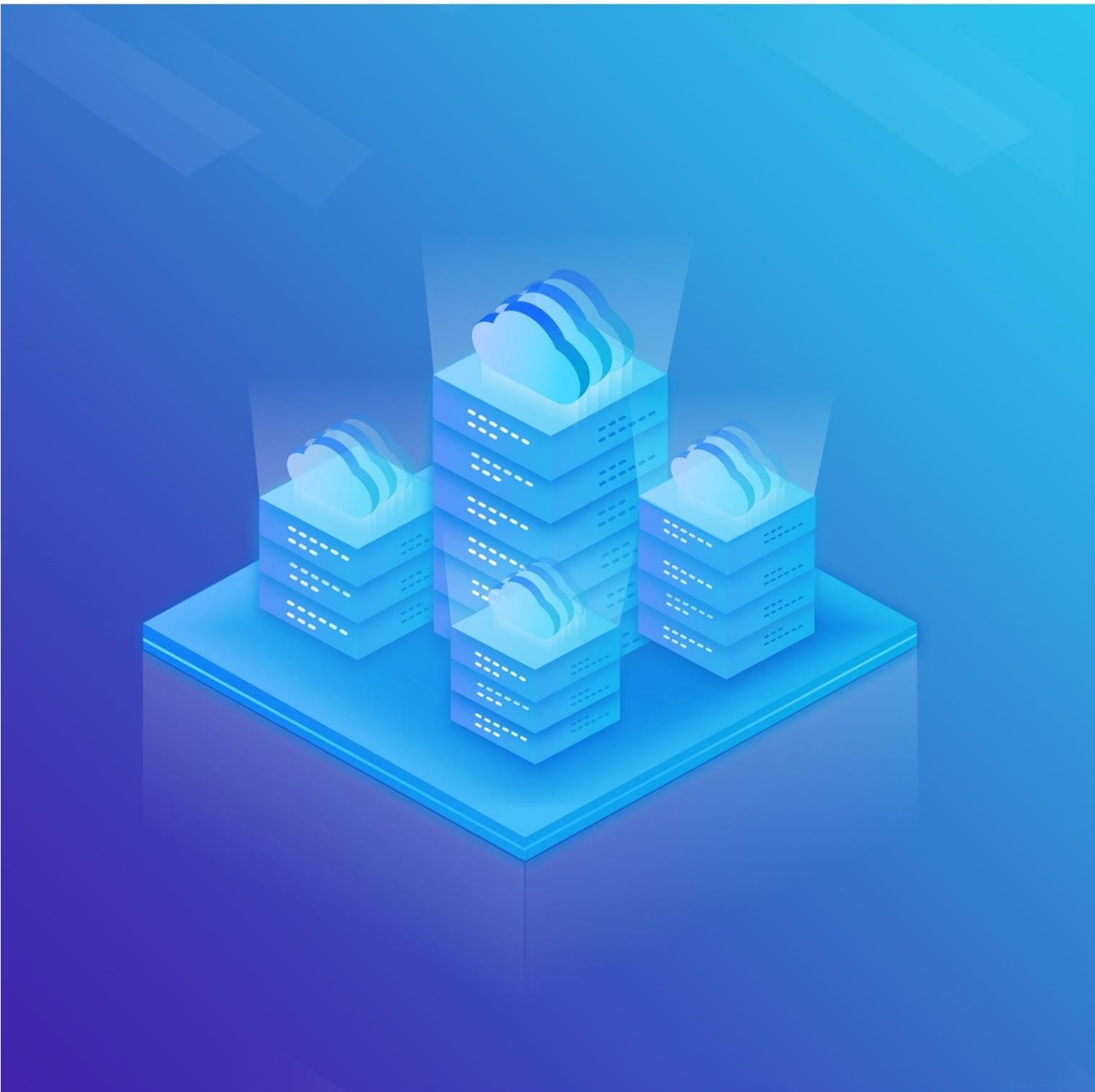
Apusic
金 蝶 天 燕

金蝶Apusic应用服务器

用户管理手册



目录

1. 关于本快用户手册

1.1. 基本介绍

本手册主要介绍如何使用金蝶 Apusic 应用服务器敏捷版（简称 AAMS）配置来使用该版本提供的新功能。

演示的敏捷版版本为 V10,下面的说明以${version}代表版本号（如果使用 9.0 版本，操作类似）。

1.2. 相关资源

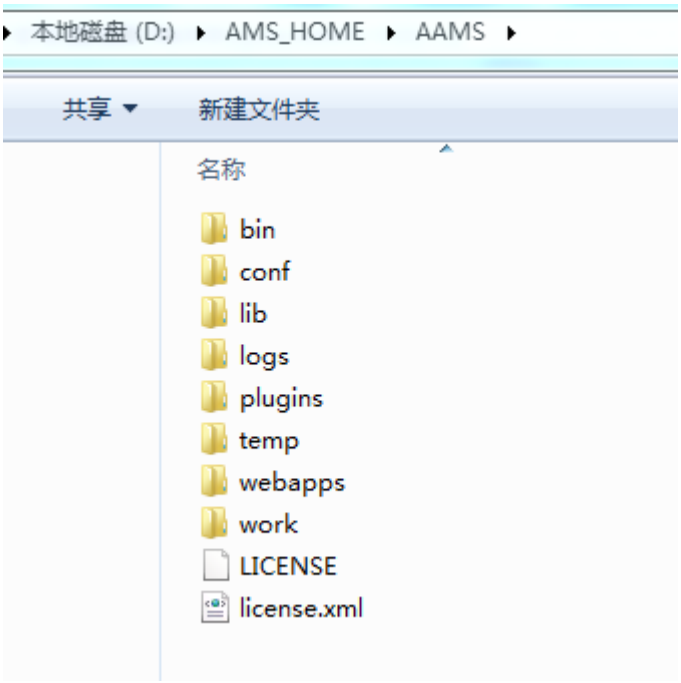
- ✓ 产品包 AAMS-${version}.zip
- ✓ 用于 SpringBoot 集成的相关包 AAMS-${version}-embed.zip
- ✓ 用于 SpringBoot 集成的 Starter 包 aams-spring-boot-starter.zip
- ✓ 集成的例子 examples.zip

2. 安装和运行

下面介绍 AAMS 作为一个完整的产品进行使用，${ AMS_HOME }表示产品安装的根目录。

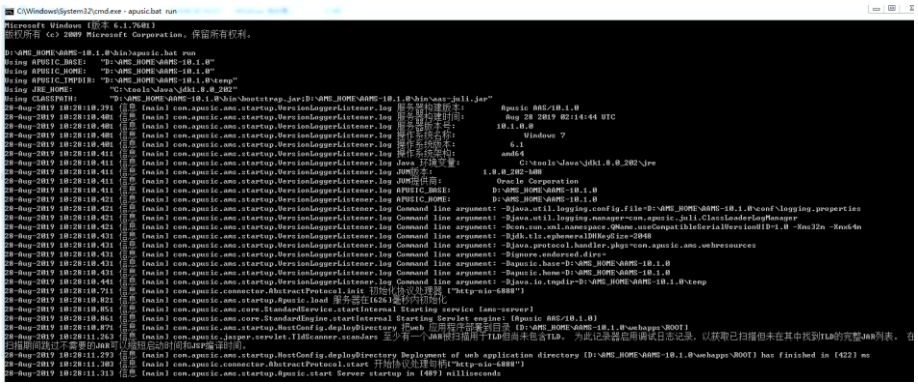
2.1. 安装

解压 AAMS-${version}.zip 到${ AMS_HOME }目录下，用${ AMS_INSTALL}目录表示其安装目录，如下图所示：



2.2. 运行

切换到 AMS 的安装的 bin 目录，打开终端，输入命令：
Windows 下输入命令：apusic.bat run
Linux 下输入命令：apusic.sh run
启动后的界面如下：



如果 Windows 启动出现乱码，则可通过修改 conf/logging.properties 文件，把控制台日志编码设置为 GBK。

```
# Handler specific properties.
# Describes specific configuration info for Handlers.
############################################################

1apusic.com.apusic.juli.AsyncFileHandler.level = FINE
1apusic.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
1apusic.com.apusic.juli.AsyncFileHandler.prefix = apusic.
1apusic.com.apusic.juli.AsyncFileHandler.maxDays = 90
1apusic.com.apusic.juli.AsyncFileHandler.encoding = UTF-8

2localhost.com.apusic.juli.AsyncFileHandler.level = FINE
2localhost.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
2localhost.com.apusic.juli.AsyncFileHandler.prefix = localhost.
2localhost.com.apusic.juli.AsyncFileHandler.maxDays = 90
2localhost.com.apusic.juli.AsyncFileHandler.encoding = UTF-8

3manager.com.apusic.juli.AsyncFileHandler.level = FINE
3manager.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
3manager.com.apusic.juli.AsyncFileHandler.prefix = manager.
3manager.com.apusic.juli.AsyncFileHandler.maxDays = 90
3manager.com.apusic.juli.AsyncFileHandler.encoding = UTF-8

4host-manager.com.apusic.juli.AsyncFileHandler.level = FINE
4host-manager.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
4host-manager.com.apusic.juli.AsyncFileHandler.prefix = host-manager.
4host-manager.com.apusic.juli.AsyncFileHandler.maxDays = 90
4host-manager.com.apusic.juli.AsyncFileHandler.encoding = UTF-8

java.util.logging.ConsoleHandler.level = FINE
java.util.logging.ConsoleHandler.formatter = com.apusic.juli.OneLineFormatter
java.util.logging.ConsoleHandler.encoding = GBK


############################################################
# Facility specific properties.
# Provides extra control for each logger.
############################################################

com.apusic.ams.core.ContainerBase.[Apusic].[localhost].level = INFO
com.apusic.ams.core.ContainerBase.[Apusic].[localhost].handlers = 2localhost.com.apusic.juli.AsyncFileHandler

com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/manager].level = INFO
com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/manager].handlers = 3manager.com.apusic.juli.AsyncFileHandler

com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/host-manager].level = INFO
com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/host-manager].handlers = 4host-manager.com.apusic.juli.AsyncFileHandler
```

启动完成后，在浏览器中输入 http://<ip>:6888 进行访问，如果是本机，则输入 http://localhost:6888,则会出现如下的界面表示成功：

3. 部署应用

3.1 自动部署

要部署应用，只需要把应用拷贝到 webapps 目录下即可自动部署。

3.2 指定路径部署

因自动部署需要把应用拷贝到 webapps 下，现提供另一种方式部署：如果应用是压缩包格式(war 格式等)，需将应用解压，比如解压到 D:/work_test/test-webapp，然后编辑 apusic.conf 文件，在 host 下新增：

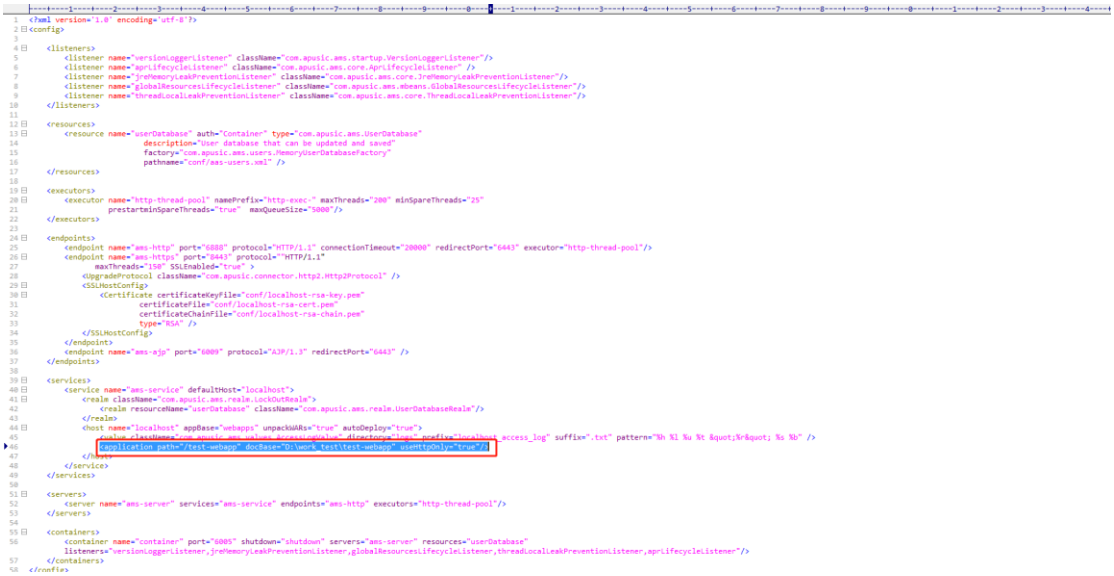
<application path="/test-webapp" docBase="D:\work_test\test-webapp" useHttpOnly="true"/>

说明：

path：部署应用的上下文路径

docBase：解压好的应用的位置

useHttpOnly：设置为 true 表示防止客户端脚本读取 JSESSIONID，防止 CSRF/XSS 恶意攻击



4. 设置 JVM 内存

Windows 下修改 apusic.bat 文件，找到 MEMORY_JVMOPTS 进行设置。

Linux 下修改 apusic.sh 文件，找到 MEMORY_JVMOPTS 进行设置。

如下图：

```
rem Add aas-juli.jar to classpath
rem aas-juli.jar can be over-ridden per instance
if not exist "%APUSIC_BASE%\bin\aas-juli.jar" goto juliClasspathHome
set "CLASSPATH=%CLASSPATH%;%APUSIC_BASE%\bin\aas-juli.jar"
goto juliClasspathDone
:juliClasspathHome
set "CLASSPATH=%CLASSPATH%;%APUSIC_HOME%\bin\aas-juli.jar"
:juliClasspathDone

if not "%JSSE_OPTS%" == "" goto gotJsseOpts
set JSSE_OPTS="-Djdk.tls.ephemeralDHKeySize=2048"
set OTHER_OPTS="-Dfile.encoding=UTF-8"
:gotJsseOpts
set "JAVA_OPTS=%JAVA_OPTS% %JSSE_OPTS% %OTHER_OPTS%"

rem Register custom URL handlers
rem Do this here so custom URL handles (specifically 'war:...') can be used in the security policy
set "JAVA_OPTS=%JAVA_OPTS% -Djava.protocol.handler.pkgs=com.apusic.ams.webresources"

rem Set by etcd etc.
set MEMORY_JVMOPTS="-Xmx4g -Xms4g -XX:NewRatio=2 -XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=256m"
if "%MEMORY_JVMOPTS%" == "" goto skipMemOpts
set "JAVA_OPTS=%JAVA_OPTS% %MEMORY_JVMOPTS%"
:skipMemOpts

set OTHERS_JVMOPTS=""
if "%OTHERS_JVMOPTS%" == """" goto skipOthOpts
set "JAVA_OPTS=%JAVA_OPTS% %OTHERS_JVMOPTS%"
:skipOthOpts

if not "%LOGGING_CONFIG%" == "" goto noJuliConfig
set LOGGING_CONFIG=-Dnop
if not exist "%APUSIC_BASE%\conf\logging.properties" goto noJuliConfig
set LOGGING_CONFIG=-Djava.util.logging.config.file="%APUSIC_BASE%\conf\logging.properties"
:noJuliConfig

if not "%LOGGING_MANAGER%" == "" goto noJuliManager
set LOGGING_MANAGER=-Djava.util.logging.manager=com.apusic.juli.ClassLoaderLogManager
:noJuliManager

rem Configure JAVA 9 specific start-up parameters
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.base/java.lang=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.base/java.io=ALL-UNNAMED"
set "JDK_JAVA_OPTIONS=%JDK_JAVA_OPTIONS% --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED"

rem Java 9 no longer supports the java.endorsed.dirs
```

5. 配置文件说明

5.1 apusic.conf 文件

apusic.xml 是金蝶 Apusic 应用服务器的重要配置文件，可通过该文件设置端口、线程池大小、设置部署应用路径、性能调优等等。

如下图：

```
<?xml version='1.0' encoding='utf-8'?>
<config>

    <listeners>
        <listener name="versionLoggerListener" className="com.apusic.ams.startup.VersionLoggerListener"/>
        <listener name="aprLifecycleListener" className="com.apusic.ams.core.AprLifecycleListener" />
        <listener name="jreMemoryLeakPreventionListener" className="com.apusic.ams.core.JreMemoryLeakPreventionListener"/>
        <listener name="globalResourcesLifecycleListener" className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener"/>
        <listener name="threadLocalLeakPreventionListener" className="com.apusic.ams.core.ThreadLocalLeakPreventionListener"/>
    </listeners>

    <resources>
        <resource name="userDatabase"
            auth="Container"
            type="com.apusic.ams.UserDatabase"
            description="User database that can be updated and saved"
            factory="com.apusic.ams.users.MemoryUserDatabaseFactory"
            pathname="conf/aas-users.xml" />
    </resources>

    <executors>
        <executor name="http-thread-pool"
            namePrefix="http-exec-"
            maxThreads="200"
            minSpareThreads="25"
            prestartminSpareThreads="true"
            maxQueueSize="5000"/>
    </executors>

    <endpoints>
        <endpoint name="ams-http"
            port="6888"
            protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="6443"
            executor="http-thread-pool"
            maxConnections="10000"
            enableLookups="false"
            acceptCount="100"
            maxPostSize="104857600"
            maxHttpHeaderSize="8192"
            compression="on"
            disableUploadTimeout="true"
            compressionMinSize="2048"
            compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf"
            URIEncoding="utf-8"
            processorCache="20000"
            tcpNoDelay="true"
            keepAliveTimeout="15000"
            maxKeepAliveRequests="100">
        </endpoint>
    </endpoints>

    <services>
        <service name="ams-service" defaultHost="localhost">
            <realm className="com.apusic.ams.realm.LockOutRealm">
                <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
            </realm>
            <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
                <application path="/test-webapp" docBase="D:\work_test\test-webapp" useHttpOnly="true"/>
            </host>
        </service>
    </services>

    <servers>
        <server name="ams-server" services="ams-service" endpoints="ams-http" executors="http-thread-pool"/>
    </servers>

    <containers>
        <container name="container"
            port="-1"
            shutdown="shutdown"
            servers="ams-server"
            resources="userDatabase"
            listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"/>
    </containers>
</config>

```

5.1.1 listeners 配置（监听器）

（1）VersionLoggerListener：启动时控制台打印应用服务器版本信息

（2）AprLifecycleListener：使用 APR 进行接收客户端请求，在应用服务器初始化之前、初始化之前尝试初始化 APR 库，成功则使用 APR 接受处理客户端请求，应用服务器销毁之后，该监听器会做 APR 的清理工作。

（3）JreMemoryLeakPreventionListener：防止 JRE 内存泄露

（4）GlobalResourcesLifecycleListener：初始化化 JNDI 资源的 MBean

（5）ThreadLocalLeakPreventionListener：防止 ThreadLocal 对象带来的内存泄漏

5.1.2 resources 配置（用户权限）

userDatabase：用户权限配置，具体可见 aas-user.xml 文件

5.1.3 executors 配置（线程池）

```
<executor name="http-thread-pool"
          namePrefix="http-exec-"
          maxThreads="200"
          minSpareThreads="25"
          prestartminSpareThreads="true"
          maxQueueSize="5000"/>
```

说明：

name：线程池名称，必须唯一，上面的是 http 的线程池配置，性能调优可进行调整

namePrefix：线程池里面的线程的前缀

maxThreads：最大线程数

minSpareThreads：最小空闲线程数

prestartminSpareThreads：是否预先启动最小空闲线程

maxQueueSize：线程池队列最大任务数

5.1.4 endpoints 配置（网络 IO 和协议）

```
<endpoint  name="ams-http"
                port="6888"
                protocol="HTTP/1.1"
                connectionTimeout="20000"
                redirectPort="6443"
                executor="http-thread-pool"
                maxConnections="10000"
                enableLookups="false"
                acceptCount="100"
                maxPostSize="104857600"
                maxHttpHeaderSize="8192"
                compression="on"
                disableUploadTimeout="true"
                compressionMinSize="2048"
                compressableMimeType="text/html,text/plain,text/css,application/javascript,
application/json,application/x-font-ttf,application/x-font-otf"
                URIEncoding="utf-8"
                processorCache="20000"
                tcpNoDelay="true"
                keepAliveTimeout="15000"
                maxKeepAliveRequests="100">
          </endpoint>
```

说明：

name：名称

port：监听端口

protocol：处理协议，可选的有 HTTP/1.1、AJP/1.3

connectionTimeout：连接超时时间

redirectPort：如果是 https，重定向端口

executor：使用的线程池

maxConnections：最大并发数

enableLookups：是否启用 DNS 查询

acceptCount：backlog 大小

maxPostSize：最大提交数据，单位字节

maxHttpHeaderSize：最大提交 http header 个数

compression：是否启用压缩

disableUploadTimeout：是否禁用上传超时

compressionMinSize：压缩文件阀值，小于该值不会进行压缩

compressableMimeType：可压缩的类型

URIEncoding：配置 url 中参数编码

processorCache：缓存 Processor 的最大数量

tcpNoDelay：是否禁用了 Nagle 算法，允许小包的发送

keepAliveTimeout：keepalive 的超时时间

maxKeepAliveRequests：保持 keepalive 的连接的个数，超过就会关闭连接

5.1.5 services 配置（虚拟主机和应用）

```
<service name="ams-service" defaultHost="localhost">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatab
aseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
        <application path="/test-webapp" docBase="D:\work_test\test-webapp" useHtt
pOnly="true"/>
    </host>
</service>
```

说明：

service：配置虚拟主机和引用用户配置

name：服务名称，必须全局唯一

defaultHost：默认虚拟主机名称，

realm：用于处理登录和登出相关信息

host：虚拟主机

name：虚拟名称，必须全局唯一

appBase：应用部署路径

unpackWARs：是否解压 war 进行部署

autoDeploy：是否自动部署

application：指定路径部署应用

path：上下文访问路径

docBase：解压的应用路径

useHttpOnly：设置为 true 表示防止客户端脚本读取 JSESSIONID，防止 CSRF/XSS 恶意攻击

5.1.6 servers 配置（引用其他功能）

```
<servers>
    <server name="ams-server" services="ams-service" endpoints="ams-http" executors="http-thread-pool"/>
</servers>
```

说明：

server：把多个服务、网络和协议、线程池等功能整合。

name：全局唯一

services：引用的服务，多个用逗号隔开

endpoints：引用的网络和协议配置，多个用逗号隔开

executors：引用的线程池，多个用逗号隔开，注意：该 server 里面的 endpoints 里面引用的线程池也必须在这里引用，如上面的 http-thread-pool

5.1.7 containers 配置（监听器等）

```
<container  name="container"
            port="-1"
            shutdown="shutdown"
            servers="ams-server"
```

resources="userDatabase"
listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"/>

说明：

container：应用服务器容器实例，可配置属性有：

name：名称，必须全局唯一。

port：侦听端口，用于关闭应用服务器，不使用可设置为-1，建议不使用。

servers：该应用服务器实例引用的服务器，多个可用逗号隔开，建议使用单个。

resources：该应用服务器实例引用的资源，比如用户名密码等，多个可用逗号隔开。

listeners：该应用服务器实例引用的监听器，多个可用逗号隔开。引用的是文件开头配置的监听器。

5.2 其他配置

5.2.1 共享库配置

共享库可以在多个应用中共享相同的类库，降低内存的使用。配置的步骤如下：

可以在${ AMS_HOME }/lib 目录下创建一个目录，如 sharelib;

目录中放置需要在多个应用中共享的 jar 包;

配置${ AMS_HOME }/conf 下的 apusic.properties，修改 shared.loader 的配置为：shared.loader="${apusic.home}/lib/sharelib/*.jar"

经过上面的步骤即可完成共享库配置，应用则可以直接使用该 jar 的资源，共享库的类库优先级低于应用中相同的类库优先级。

5.2.2 手动部署多个应用

在普通方式部署时，应用除了直接拷贝到 webapps 目录自动进行部署外，也可以通过修改配置文件手动部署：修改 apusic.conf 文件中的 host 标签，增加如下的类似配置，其中部署了 2 个 application,属性 path 为指定的上下文路径，docBase 是应用的目录：

<host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true

">

<application path="/test2" docBase="H:/aa/test1.war" useHttpOnly="true"/>

<application path="/test3" docBase="H:/aa/test1.war" useHttpOnly="true"/>

</host>

5.2.3 内存的修改

修改 apusic.sh 或 apusic.bat 中的变量 MEMORY_JVMOPTS 即可。

5.2.4 端口的修改

修改 apusic.conf 文件中 endpoint name="ams-http"的 port 属性，默认为 port="6888"

5.2.5 开启 JMX

JMX 服务默认为不开启，如果需要开启 JMX 服务，首先需要检查配置文件中是否存在 ams-jmx 服务配置，如下：

<endpoint name="ams-jmx"

port="6868"

auth="false"

protocol="JMXRMI/2.0">

</endpoint>

如果没有则需要添加上。再需要在 apusic.conf 中找到如下的配置，把 ams-jmx 增加到 endpoints 属性中，如下配置：

<server name="ams-server" services="ams-service" endpoints="ams-http,ams-jmx" executors="http-thread-pool"/>

JMX 默认访问不进行授权验证，如果需要授权认证，则需要修改相关配置并制定授权文件：

修改 ams-jmx 服务属性 auth 设置为 true，配置密码和访问授权文件属性 passFile 和 accessFile，默认为 conf 目录下的 jmxremote.password 和 jmxremote.access 文件

jmxremote.password 文件内容类似：

controlRole　weiys

其中 controlRole 为用户名，weiys 为密码

jmxremote.access 文件内容类似：

controlRole　readonly

controlRole　readwrite \

create javax.management.monitor.*,javax.management.timer.* \

unregister

5.2.6 启动和停止应用

如果 JMX 服务开启，则可以通过 JMX 启动和停止应用，如下，其中 appName 的参数值是应用的名字，port 参数的值是 JMX 端口，现在只支持在本地运行：

启动：./startapusic.sh startapp --appName test1

停止：./startapusic.sh stopapp --appName test1

如果开启了安全认证，则需要指定用户名和密码：

启动命令类似：startapusic.bat startapp --appName test1 --userName controlRole --use rPass weiys

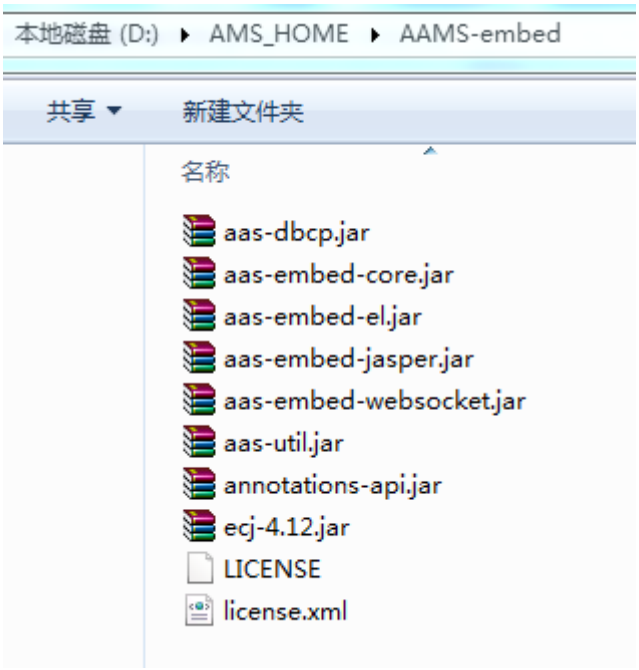
如果一次要启动或停止多个应用，则 appName 可以输出多个值，如 test1 test2

6. 与 Springboot 集成使用

例子使用了 maven 进行打包、构建和运行，所以需要先注册产品的 jar 到软件库中，以安装到本地的软件库为例进行说明。

6.1 AAMS 核心 jar 包的注册

解压 AAMS-${version}-embed.zip 到${ AMS_HOME }目录中，如下图所示，这些核心 jar 包的内容是把${AMS_INSTALL}\lib 下的核心 jar 进行了合并，只是方便引入和管理。



安装各个软件包到本地库的命令如下（在当前目录打开终端，输入如下的命令）：

mvn install:install-file -Dfile=aas-embed-core.jar -DgroupId=com.apusic.ams.embed -DartifactId=aas-embed-core -Dversion=10.1 -Dpackaging=jar

mvn install:install-file -Dfile=aas-embed-el.jar -DgroupId=com.apusic.ams.embed -DartifactId=aas-embed-el -Dversion=10.1 -Dpackaging=jar

mvn install:install-file -Dfile=aas-embed-websocket.jar -DgroupId=com.apusic.ams.embed -DartifactId=aas-embed-websocket -Dversion=10.1 -Dpackaging=jar

mvn install:install-file -Dfile=aas-embed-jasper.jar -DgroupId=com.apusic.ams.embed -DartifactId=aas-embed-jasper -Dversion=10.1 -Dpackaging=jar

mvn install:install-file -Dfile=aas-dbcp.jar -DgroupId=com.apusic.ams.embed -DartifactId=aas-dbcp -Dversion=10.1 -Dpackaging=jar

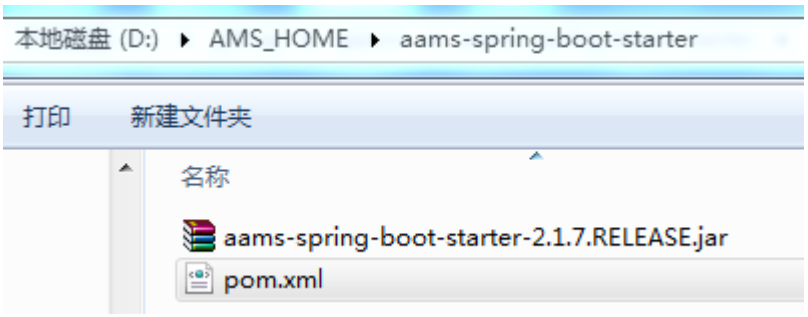
mvn install:install-file -Dfile=aas-util.jar -DgroupId=com.apusic.ams.embed -DartifactId=aas-util -Dversion=10.1 -Dpackaging=jar

mvn install:install-file -Dfile=ecj-4.12.jar -DgroupId=com.apusic.ams.embed -DartifactId=ecj -Dversion=4.12 -Dpackaging=jar

该产品包的 license.xml 文件在下面运行案例时会使用到。

6.2 Starter 的注册

该版本是支持 spring2.X 的 Starter。解压 aams-spring-boot-starter.zip 到${ AMS_HOME } 中，如下图：



安装 starter 到本地库，方便进行使用，命令类似如下（在当前目录打开终端，输入如下的命令）：

mvn install:install-file -Dfile=aams-spring-boot-starter-2.1.7.RELEASE.jar -DgroupId=com.apusic -DartifactId=aams-spring-boot-starter -Dversion=2.1.7.RELEASE -Dpackaging=jar -DpomFile=pom.xml

说明：如果需要引入其他的${AMS_INSTALL}\lib 下的核心 jar，可以修改这里的 pom.xml 文件，重新运行上面的命令进行注册。

6.3 参数配置

通过修改 application.properties，可配置如下的参数：

server.port: 8080　　应用服务器监听端口

server.servlet.context-path: /demo　　应用访问的上下文

server.aas.max-threads=500　　最大线程数

server.aas.min-spare-threads=64　　最小空闲线程数

server.aas.max-http-header-size=8192　　最大支持请求头个数

server.aas.max-http-post-size=8192　　最大 post 参数个数

server.aas.connection-timeout=60000　　建立连接超时时间

server.aas.max-connections=20000　　最大并发个数

server.aas.accept-count=1024　　已连接但应用服务器未处理数

server.aas.max-swallow-size=20971520　　请求体大小限制

server.aas.compression=true　　是否启用压缩

server.aas.compression-min-size=4096　　超过该大小的文件才启用压缩

server.aas.compressable-mime-type="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,application/xml"　　需要压缩的文件类型

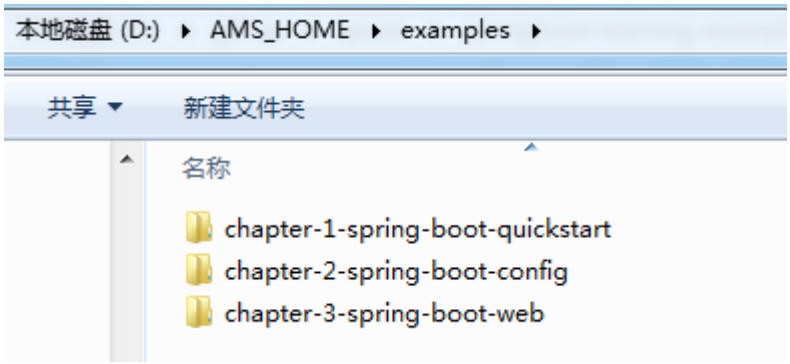
server.aas.max-keep-alive-timeout=15000　　连接保活超时时间

server.aas.max-keep-alive-requests=-1　　最多处理多少个请求后关闭连接，-1 表示不限制

server.aas.disable-upload-timeout=true　　是否禁用上传超时

6.4 例子的使用

解压 examples.zip 到${AMS_HOME},里面有 3 个使用例子，如下图：



6.3.1 例子 1-chapter-1-spring-boot-quickstart

在这个例子中，我们演示如何使用 AAMS 作为 Springboot 应用的运行容器

1. Pom 文件的变化

这个例子中，要使用 Apusic 作为运行容器，有 2 个地方要进行改动：

1) 由于 spring-boot-starter-web 插件默认带了 tomcat 的 starter，所以在 pom.xml 中需要排除 tomcat 的 starter，如下面的配置：

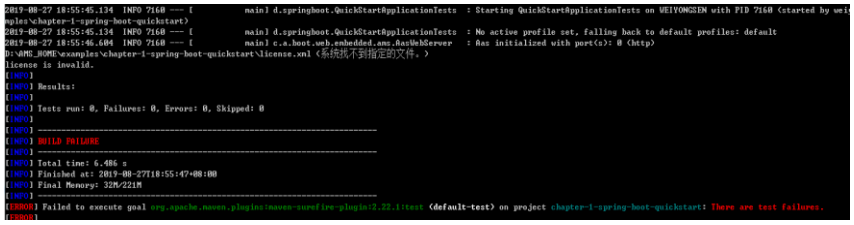
```
<!-- Web 依赖 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-tomcat</artifactId>
                </exclusion>
            </exclusions>
    </dependency>
```

2) 增加 aams 的 starter 依赖,如 pom.xml 中的如下描述：

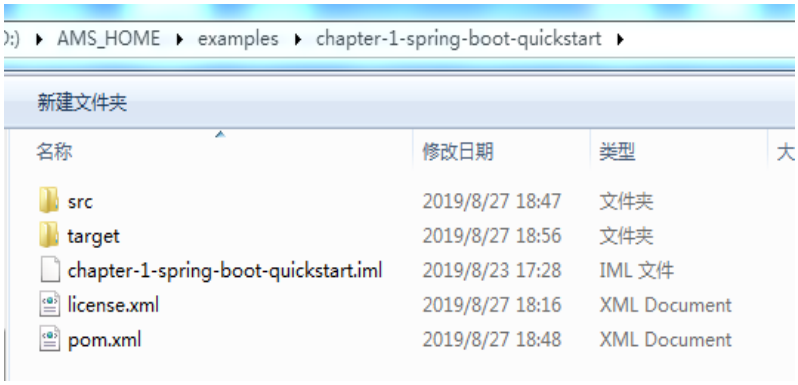
```
<dependency>
        <groupId>com.apusic</groupId>
        <artifactId>aams-spring-boot-starter</artifactId>
        <version>2.1.7.RELEASE</version>
</dependency>
```

2. 构建产品包

进入到工程目录下，并打开终端，并输入 mvn package 命令进行构建。在构建的过程中要进行测试，需要启动 ams，发现出现提示如下信息后失败，如下图：



发生该问题是由于在当前目录没有授权文件，这时把之前 AMS 产品包目录下的 license.xml 文件拷贝到该目录下，再进行构建则构建成功。

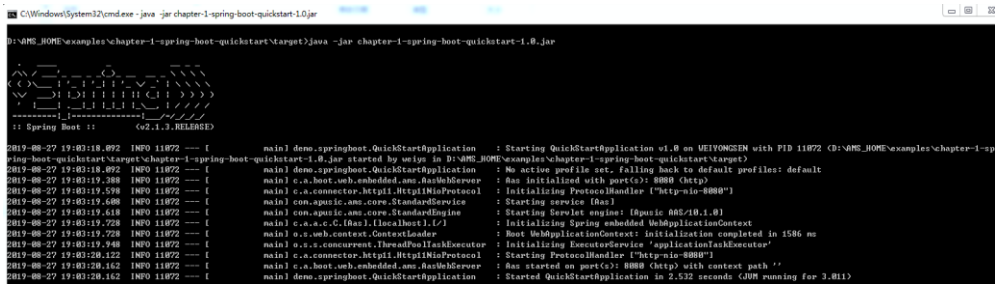


3. 运行产品例子

切换到构建后的 target 目录，在该目录下打开终端，并运行如下命令进行运行（由于当前目录在 target 下，需要把 license.xml 文件拷贝到 target 目录下）：

java –jar chapter-1-spring-boot-quickstart-1.0.jar

启动后类似如下的界面：



启动成功后，在浏览器中输入：http://localhost:8080/hello，先如下界面则表示成功：

其他：也可以使用命令 mvn spring-boot:run 直接进行启动。

6.3.2 例子 2- chapter-2-spring-boot-config

修改 pom.xml 的文件，加入 AAMS 的 starter，修改方式同例子 1，构建和运行的方式和例子 1 页类似。

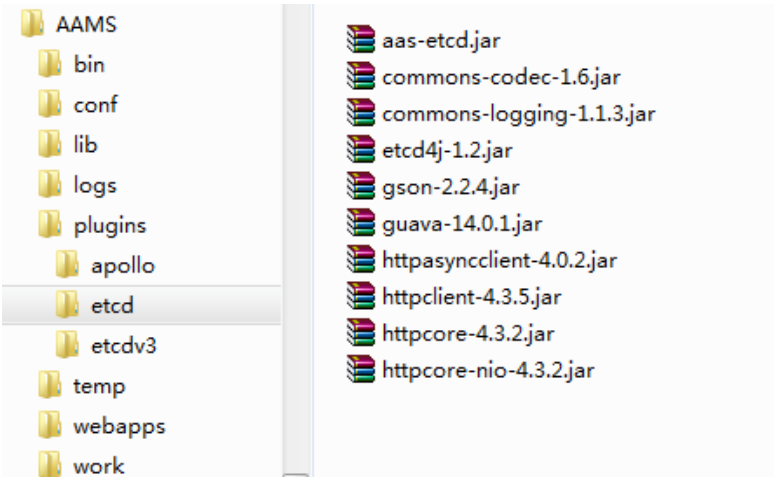
在例子 2 的目录下打开终端，输入命令：

mvn spring-boot:run

启动完成后，在本机浏览器中访问 http://localhost:8080/book/hello 会出现如下界面，则表示成功。

7. 集中配置说明

该版本可以支持集中配置软件包括 ETCD 和 apollo，后续会增加更多的支持。对集中配置软件的实现包，放在产品包的 plugins 目录下，如下面图所示：



备注：一般情况下一个用户场景只会使用到一个实现，可以删除其他的实现，以节约空间。

7.1 使用 apollo 集中配置

下面介绍如何在应用服务器中使用 apollo 软件进行集中配置。

7.1.1 添加集中配置服务

apusic.conf 配置文件需要增加 plugins 的配置，并指定 apollo 的实现，如下图所示：

```
<plugins>
    <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader"
cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
</plugins>
```

其中 className="com.apusic.ams.config.listener.ApolloPluginLoader" 是加载 apollo 配置的类，属性包括了：

appId（必填） 配置的应用 id
cluster（选填）配置属于的集群名称
metaUrl（必填） Apollo 提供服务的地址
namespace（选填）配置的命名空间
env（选填） 配置的环境（开发环境，正式环境等等）
idc（选填） 配置的数据中心

以上属性可通过配置文件配置，也可通过环境变量配置，也可通过系统变量设置

优先级是：环境变量 > 系统变量 > 配置文件

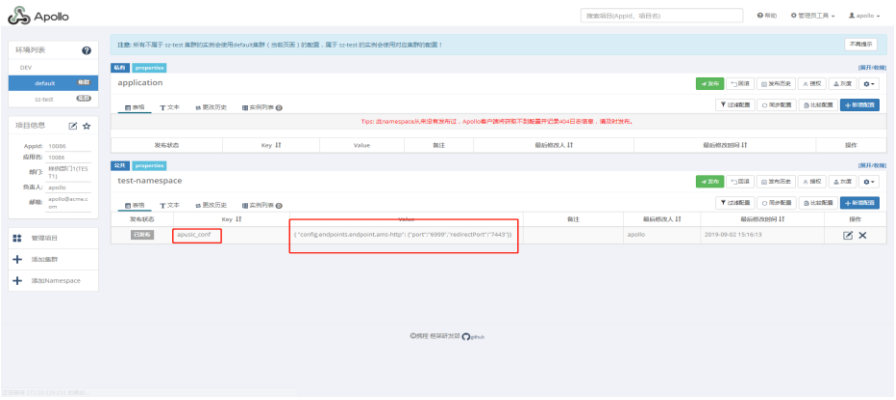
一般环境变量和系统变量的变量名称为大写的，环境变量(System.getProperty("xxx")获取)和系统变量(System.getEnv("xxx")获取)的变量名为如下，每个变量名会与上面的属性对应：

AAS_CONFIG_APOLLO_APP_ID
AAS_CONFIG_APOLLO_META_SERVER
AAS_CONFIG_APOLLO_CLUSTER
AAS_CONFIG_APOLLO_IDC
AAS_CONFIG_APOLLO_NAMESPACE
AAS_CONFIG_APOLLO_ENV

7.1.2 在 apollo 的配置

1. apusic.conf 配置

与配置中心整合后，应用服务器的配置文件（apusic.conf 文件）可通过设置 key 为 apusic_config，value 为要改变的值进行修改，如下图：



其中 key 必须为 apusic_config，value 必须是 json 字符串。

```
{
    "config.endpoints.endpoint.ams-http": {
        "port": "6999",
        "redirectPort": "7443"
    }
}
```

config.endpoints.endpoint.ams-http 拆分对应这 config > endpoints > endpoint > ams-http 对应这 xml 的层级结构：

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>

<config>
    <plugins>
        <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
    </plugins>

    <listeners>
        <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
        <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
        <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
        <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
        <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
    </listeners>

    <resources>
        <resource auth="Container" description="User database that can be updated and saved" factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase"
        pathname="conf/aas-users.xml" type="com.apusic.ams.UserDatabase"/>
    </resources>

    <executors>
        <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-" prestartminSpareThreads="true"/>
    </executors>

    <endpoints>
        <endpoint URIEncoding="utf-8" acceptCount="100"
        compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image
        /png,image/gif,audio/mpeg,video/mp4" compression="on" compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
        executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192" maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http"
        port="6999" processorCache="20000" protocol="HTTP/1.1" redirectPort="7443" tcpNoDelay="true">
        </endpoint>
    </endpoints>

    <services>
        <service defaultHost="localhost" name="ams-service">
            <realm className="com.apusic.ams.realm.LockOutRealm">
                <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
            </realm>
            <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">

            </host>
        </service>
    </services>

    <servers>
        <server endpoints="ams-http" executors="http-thread-pool" name="ams-server" services="ams-service"/>
    </servers>

    <containers>
        <container listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"
        name="container" port="-1" resources="userDatabase" servers="ams-server" shutdown="shutdown"/>
    </containers>
</config>
```

port 和 redirectPort 对应着属性。

```
{
    "config.endpoints.endpoint.ams-http": {
        "port": "6999",
        "redirectPort": "7443"
    }
}
```

上图的意思就是，修改名称为 ams-http 的 xml 标签为 endpoint 的 port 属性的值为 6999，redirectPort 的属性值为 7443

2. 数据源的配置

通过 apollo 软件的操作界面进行配置，数据的格式可以参考 ETCD 章节下的说明。

3. 日志的配置

通过 apollo 软件的操作界面进行配置，数据的格式可以参考 ETCD 章节下的说明。

4. JVM 的配置

通过 apollo 软件的操作界面进行配置，数据的格式可以参考 ETCD 章节下的说明。

7.2 使用 ETCD 集中配置

7.2.1 添加集中配置服务

apusic.conf 配置文件需要增加 plugins 的配置，并指定 etcd 的配置实现。

要支持 ETCD2，则需要指定实现为：

<plugins>

<plugin className="com.apusic.ams.config.listener.EtcdPluginLoader" clusterId="cluster" host="172.18.100.74" port="2379" namespace="test"/>

</plugins>

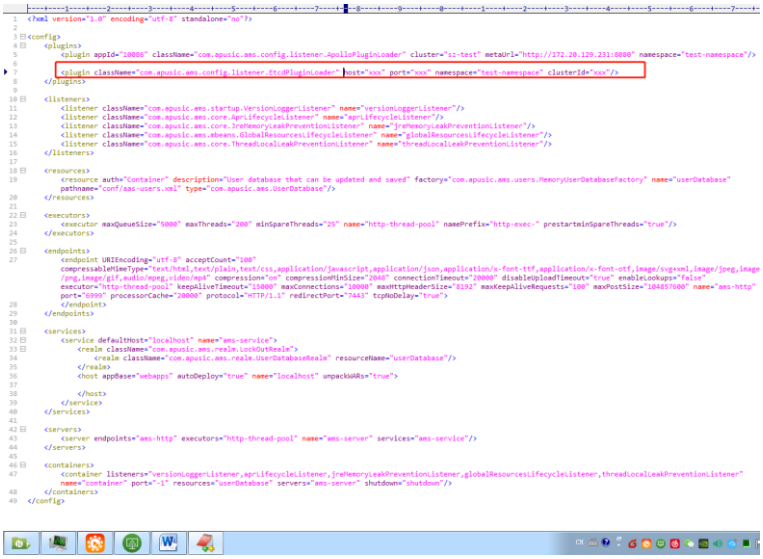
要支持 ETCD3 则需要指定为：

<plugins>

<plugin className="com.apusic.ams.config.listener.EtcdV3PluginLoader" clusterId="cluster" host="172.18.100.74" port="2379" namespace="test"/>

</plugins>

示例如下图所示：



其中 className="com.apusic.ams.config.listener.EtcdPluginLoader" 是加载 etcd 配置的类，属性包括了属性说明：

host ： 安装 etcd 的服务器 ip 地址,必须
port：etcd 服务的客户端访问端口，必须
namespace：应用的命名空间，必须
clusterId：应用的集群标识，必须

以上属性可通过配置文件配置，也可通过环境变量配置，也可通过系统变量设置

优先级是：环境变量 > 系统变量 > 配置文件

一般环境变量和系统变量的变量名称为大写的，环境变量(System.getProperty(“xxx”)获取)和系统变量(System.getEnv(“xxx”)获取)的变量名为如下，每个变量名会与上面的属性对应：

AAS_CONFIG_ETCD_HOST

AAS_CONFIG_ETCD_PORT

AAS_CONFIG_ETCD_NAMESPACE

AAS_CONFIG_ETCD_CLUSTERID

7.2.2 在 ETCD 的配置

1. apusic.conf 的配置

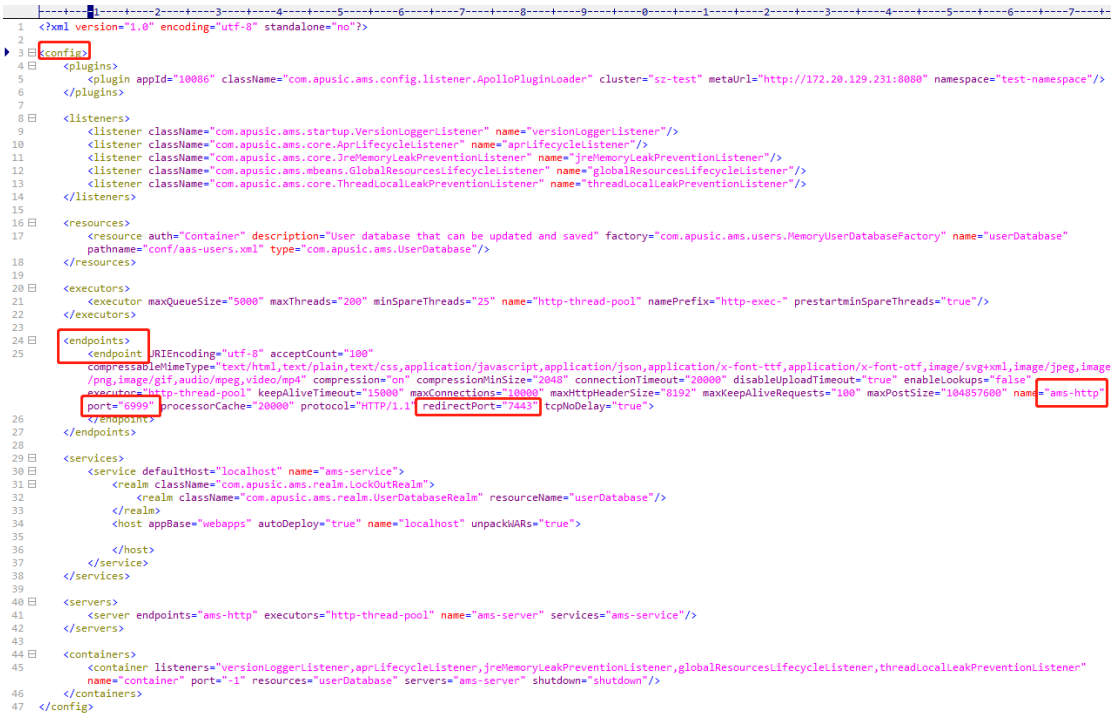
与配置中心整合后，应用服务器的配置文件（apusic.conf 文件）可通过设置 key 为 apusic_config，value 为要改变的值，进行修改。

在 ETCD 服务器中设置如下的值，则表示修改端口为 6999 和 7443：

./etcdctl put /test/cluster/aamsconfig/apusic_config '{"config.endpoints.endpoint.ams-http":{"port":"6999","redirectPort":"7443"}}'

配置信息最后保存在 conf 目录下的 apusic.conf 文件。

config.endpoints.endpoint.ams-http 拆分对应这 config > endpoints > endpoint > ams-http 对应这 xml 的层级结构：



```
<?xml version="1.0" encoding="utf-8" standalone="no"?>

<config>
    <plugins>
        <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
    </plugins>

    <listeners>
        <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
        <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
        <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
        <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
        <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
    </listeners>

    <resources>
        <resource auth="Container" description="User database that can be updated and saved" factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase"
        pathname="conf/aas-users.xml" type="com.apusic.ams.UserDatabase"/>
    </resources>

    <executors>
        <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-" prestartminSpareThreads="true"/>
    </executors>

    <endpoints>
        <endpoint URIEncoding="utf-8" acceptCount="100"
        compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image
        /png,image/gif,audio/mpeg,video/mp4" compression="on" compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
        executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192" maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http"
        port="6999" processorCache="20000" protocol="HTTP/1.1" redirectPort="7443" tcpNoDelay="true">
        </endpoint>
    </endpoints>

    <services>
        <service defaultHost="localhost" name="ams-service">
            <realm className="com.apusic.ams.realm.LockOutRealm">
                <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
            </realm>
            <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">

            </host>
        </service>
    </services>

    <servers>
        <server endpoints="ams-http" executors="http-thread-pool" name="ams-server" services="ams-service"/>
    </servers>

    <containers>
        <container listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"
        name="container" port="-1" resources="userDatabase" servers="ams-server" shutdown="shutdown"/>
    </containers>
</config>
```

port 和 redirectPort 对应着属性。

```
{
    "config.endpoints.endpoint.ams-http": {
        "port": "6999",
        "redirectPort": "7443"
    }
}
```

上图的意思就是，修改名称为 ams-http 的 xml 标签为 endpoint 的 port 属性的值为 6999，redirectPort 的属性值为 7443

2. 数据源的配置

与配置中心整合后，数据源配置的 key 为 datasource_config，它是一个 json 格式的字符串。可以在 etcd 中增加如下的配置：

./etcdctl put /test/cluster/aamsconfig/datasource_config '{"jdbc/test":{"auth":"Container","type":"javax.sql.DataSource","maxActive":"100","username":"root","password":"root","driverClassName":"com.mysql.jdbc.Driver","url":"jdbc:mysql://172.20.129.76:3306/test","maxIdle":"30","maxWait":"10000"}}'

其中 jdbc/test 为数据源的名称，如果存在该名称，则进行修改动作，如果不存在该名称，则进行添加的动作。配置信息最后保存在 conf 目录下的 context.xml 文件。

3. 日志的配置

与配置中心整合后，日志的配置的 key 为 logging_conf，它是一个 json 格式的字符串。可以在 etcd 中增加如下类似的配置，修改多个配置，则增加多个 json 格式的 key-value：

./etcdctl put /test/cluster/aamsconfig/datasource_config '{"com.apusic.connector.http2.level":"FINER"}'

配置信息最后保存在 conf 目录下的 logging.properties 文件

4. JVM 的配置

与配置中心整合后，日志的配置的 key 为 jvm_opt，它是一个普通格式的字符串，不是一个 json 字符串，配置的格式类似如下：

./etcdctl put /test/cluster/aamsconfig/jvm_opt '-server -Djava.net.preferIPv4Stack=true -Xms513m -Xmx1023m -XX:MaxPermSize=257m'

配置信息最后保存在 bin 目录下的 apusic.bat（window 下）或 apusic.sh(linux 下)文件。

8. 国密配置说明

8.1 环境说明

需要使用 JDK8 版本和国密浏览器(如 360 国密浏览器)。

8.2 应用服务器正书配置

8.2.1 单向认证

修改 apusic.conf 文件，增加如下的 endpoint 配置(如果已经存在，则进行修改)：

```
<endpoint name="https" port="6887" maxThreads="150" SSLEnabled="true">
        <SSLHostConfig sslProtocol="SMv1.1" protocols="SMv1.1" >
            <Certificate certificateKeystoreFile="conf/test016.pfx"
certificateKeystorePassword="123456" certificateKeyPassword="123456"
                type="EC" certificateKeystoreType="PKCS12"/>
        </SSLHostConfig>
</endpoint>
```

其中属性：

certificateKeystoreFile 表示证书文件的路径

protocols 和 sslProtocol 需要指定为 SMv1.1

certificateKeystorePassword 为证书库文件的密码

certificateKeyPassword 为证书的密码


test016.pfx

8.2.2 双向认证

修改 apusic.conf 文件，增加如下的 endpoint 配置(如果已经存在，则进行修改)：

```
<endpoint name="https" port="6887" maxThreads="150" SSLEnabled="true">
        <SSLHostConfig sslProtocol="SMv1.1" protocols="SMv1.1"
truststoreFile="conf/truststore.jks" truststorePassword="123456" certificateVerification="want"
truststoreType="JKS">
            <Certificate certificateKeystoreFile="conf/keystore.p12"
certificateKeystorePassword="123456" certificateKeyPassword="123456" type="EC"
certificateKeystoreType="PKCS12"/>
</endpoint>
```

其中属性：

truststoreFile 指定信任库文件，其类型是 JKS

truststorePassword 信任库密码

certificateVerification 是否进行证书验证，双向认证时为 required 值，可选客户端认证设置为 want


keystore.p12


truststore.jks

8.2.3 增加 endpoint 到 server

修改 apusic.conf 文件，修改如下的配置,其中 endpoints 属性中的 https 就是上面配置单向认证或双向认证的 endpoint 的名字：

```
<server name="ams-server" services="ams-service" endpoints="http,https"
executors="http-thread-pool"/>
```

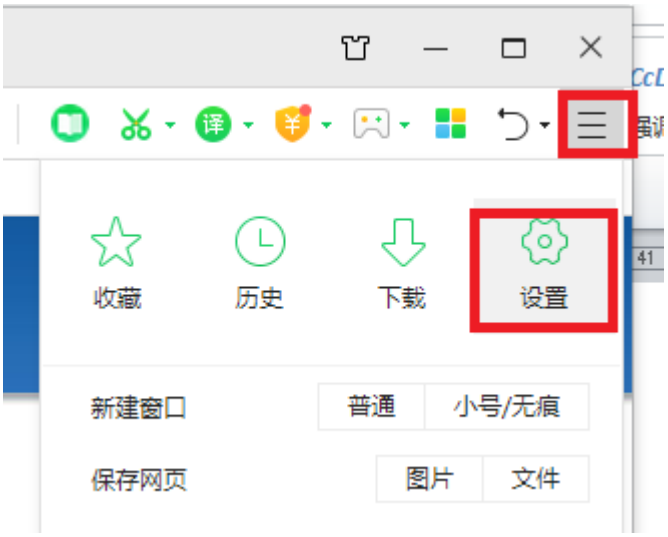
8.3 360 浏览器客户端

8.3.1 浏览器安装

从 360 网站下载国密浏览器(http://browser.360.cn/se/ver/gmzb.html)进行安装，测试使用的是 V10 版本；

8.3.2 选择支持国密

打开 360 浏览器，点击浏览器的右上角的“三”按钮，再点击“设置”选项，如下图：



然后进入”安全设置”tab 页，勾选“启用国密 ssl 协议支持”复选框，如下图：

8.3.3 根证书内容处理

Window 下的处理：

在 C:\Users\用户名\AppData\Roaming\360se6\User Data\Default\gmssl(该路径中的 360se6 目录可能有所变化如 360se 或 360ent 等)下创建 ctl 目录，然后在 ctl 目录下创建 ctl.dat 文件，并将根证书内容拷入该文件，然后重启浏览器。

备注：个人版的目录为 C:\Users\用户名\AppData\Roaming\360se\User Data\Default\gmssl

Linux 下：

linux 环境下的目录为~/.config/browser360/Default/gmssl 下创建 ctl 目录，然后在 ctl 目录下创建 ctl.dat 文件，并将根证书内容拷入该文件。然后重启浏览器。例子如下附件：


ctl.dat

8.3.4 客户端认证配置

客户端认证配置除了按照上面的”双向认证”章节配置外，还需要配置客户端证书。如果使用 ukey 作为证书，可以参考其说明文档。如海泰 key 的安装的一些问题如下：

“各平台下海泰 key 驱动相关问题： windows 下直接安装海泰 key 驱动文件即可。linux 平台下需将海泰 key 的驱动文件放入/opt/browser360/并重命名为 libSKFAPI-x64.so，将该文件的权限置为可执行权限(测试时可以用 777 模式)。”

8.4 启动国密

使用例子说明的 test016.pfx 进行服务器单向认证配置，ctl.dat 作为根证书的内容在客户端配置。修改 ams-http endpoint,增加 SSL 的配置，如下红色部分：

```
<endpoint name="ams-http"
                port="6888"
                protocol="HTTP/1.1"
                connectionTimeout="20000"
                redirectPort="6443"
                executor="http-thread-pool"
                maxConnections="10000"
                enableLookups="false"
                acceptCount="100"
                maxPostSize="104857600"
                maxHttpHeaderSize="8192"
```

```
                compression="on"
                disableUploadTimeout="true"
                compressionMinSize="2048"
                compressableMimeType="text/html "
                URIEncoding="utf-8"
                processorCache="20000"
                tcpNoDelay="true"
                keepAliveTimeout="15000"
                maxKeepAliveRequests="100" SSLEnabled="true">
        <SSLHostConfig sslProtocol="SMv1.1" protocols="SMv1.1">
                        <Certificate certificateKeystoreFile="conf/test016.pfx"
certificateKeystorePassword="123456" certificateKeyPassword="123456" type="EC"
certificateKeystoreType="PKCS12"/>
                </SSLHostConfig>
        </endpoint>
```

通过上面的步骤配置完成后，通过命令 apusic.sh gm start 或 apusic.sh gm run 可以启动国密支持的容器。启动后使用 360 国密浏览器访问 https://IP:6888 显示如下界面表示成功：

备注：可以通过在启动脚本中增加参数-Djavax.net.debug=ssl:handshake 来打印具体的 SSL 处理过程，方便分析可能存在的问题。

8.5 国密证书生成

使用 GmSSL 工具包（在相关附件章节下载）生成相关的国密证书，假设工具安装目录位${GmSSL}。具体步骤如下：

8.5.1 GmSSL 工具配置

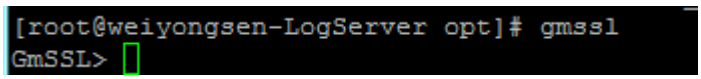
1. 把 GmSSL.tar.zip 文件拷贝的/opt 目录下，当前目录切换到/opt 目录，并通过命令 tar -xzvf GmSSL.tar.zip 进行解压，会在/opt 目录下生成 GmSSL 目录；

2. 设置下面 2 个环境变量，让 gmssl 命令可以直接运行，命令类似如下：

export PATH=$PATH:/opt/GmSSL/bin

export LD_LIBRARY_PATH=/opt/GmSSL/lib/

然后在终端输入 gmssl，能够出现输入提示符则表示成功，类似如下：



3. openssl.cnf 文件修改

该文件在/opt/GmSSL/ssl 目录下。修改的内容包括：

1) [req]选项中添加或者修改：default_md = sm3
2) 修改[v3_req]选项内容为：

```
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature
```

3) 添加[v3enc_req]选项，其内容：

```
basicConstraints = CA:FALSE
keyUsage = keyAgreement, keyEncipherment, dataEncipherment
```

8.5.2 证书制作

1. 制作前准备

创建目录/opt/test，并在目录下创建一个名为"sm2Certs"的目录；
切换当前目录到/opt/test，并拷贝 openssl.cnf 文件文件到该目录下；

2. 生成 SM2 参数文件

```
gmssl ecparam -name SM2 -out SM2.pem
```

3. 生成 ca 证书

```
gmssl req -config ./openssl.cnf -nodes -subj
"/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=Test CA (SM2)"
-keyout CA.key.pem -newkey ec:SM2.pem -new -out CA.req.pem
gmssl x509 -req -days 7300 -in CA.req.pem -extfile ./openssl.cnf
-extensions v3_ca -signkey CA.key.pem -out CA.cert.pem
rm CA.req.pem
```

4. 生成 SSL 服务端（或客户端）签名证书

```
gmssl req -config ./openssl.cnf -nodes -subj
"/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=server sign
(SM2)" -keyout sm2Certs/svrsig.key.pem -newkey ec:SM2.pem -new -out
sm2Certs/svrsig.req.pem
gmssl x509 -req -days 365 -in sm2Certs/svrsig.req.pem -CA CA.cert.pem
-CAkey CA.key.pem -extfile ./openssl.cnf -extensions v3_req -out
sm2Certs/svrsig.cert.pem -CAcreateserial
rm -f sm2Certs/svrsig.req.pem
```

5. 生成 SSL 服务端（或客户端）加密证书

```
gmssl req -config ./openssl.cnf -nodes -subj
"/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=server enc
(SM2)" -keyout sm2Certs/svrenc.key.pem -newkey ec:SM2.pem -new -out
sm2Certs/svrenc.req.pem
gmssl x509 -req -days 365 -in sm2Certs/svrenc.req.pem -CA CA.cert.pem
-CAkey CA.key.pem  -extfile ./openssl.cnf -extensions v3enc_req  -out
sm2Certs/svrenc.cert.pem -CAcreateserial
rm -f sm2Certs/svrenc.req.pem
```

6. 生成 PKCS12 秘钥库

命令中的 JSSE_GMDIR 需要根据实际的 AAS 版本和 JDK 版本进行指定，如目录 /AAMS-V10/lib/endorsedGM/JDK8。制作时提示输入密码，正确输入即可。

```
gmssl pkcs12 -export -inkey sm2Certs/svrsig.key.pem  -in
sm2Certs/svrsig.cert.pem -CAfile CA.cert.pem -chain  -out
sm2Certs/svrsig.p12 -name "svrsig"
gmssl pkcs12 -export -inkey sm2Certs/svrenc.key.pem  -in
```

```
sm2Certs/svrenc.cert.pem -CAfile CA.cert.pem -chain  -out
sm2Certs/svrenc.p12 -name "svrenc"
```

```
将分别生成的秘钥库“合并”
cp sm2Certs/svrsig.p12 sm2Certs/keystore.p12
java -Djava.endorsed.dirs=${JSSE_GMDIR}
sun.security.tools.keytool.Main -importkeystore -srckeystore
sm2Certs/svrenc.p12 -srcstoretype PKCS12 -destkeystore
sm2Certs/keystore.p12 -deststoretype PKCS12
```

7. 生成信任库（可选）

生成 jks 格式的信任库,JSSE_GMDIR 需要根据实际的 AAS 版本和 JDK 版本进行指定，如目录/AAMS- V10/lib/endorsedGM/JDK8，制作时提示输入密码，正确输入即可。

```
java -Djava.endorsed.dirs=${JSSE_GMDIR}
sun.security.tools.keytool.Main -importcert -trustcacerts -alias root
-file CA.cert.pem  -keystore truststore.jks -storetype  jks
```

8.5.3 证书使用

经过上面步骤后，生成了 CA 证书文件 CA.cert.pem，信任库文件 truststore.jks，服务器证书文件 keystore.p12(在 sm2Certs 目录)。

根据上面的双向认证配置章节，在服务器中使用信任库文件和服务器证书文件进行配置，而且国密浏览器中，则需要把 CA 证书文件拷贝到 ctl.dat 文件中，具体参考 360 浏览器客户端章节。

8.5.4 相关附件


keystore.p12


truststore.jks


CA.cert.pem


GmSSL.tar.zip

8.6 其他问题

如果使用 Oracle JDK，由于 JVM 默认的加密字节长度限制，需要将 lib/jce_policy-8.zip 解压，并替换%JAVA_HOME%/jre/lib/security 文件夹下相应 jar 文件，解放字节限制。

jce_policy-8.zip

9. 配置数据源

数据库连接池配置（在 context.xml 文件中新增内容，如下图）：

```
<Resource  name="jdbc/test"
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://127.0.0.1:3306/test_platform?useUnicode=true&amp;characterEncoding=utf-8"
    username="root"
    password="root"
    maxTotal="100"
    maxIdle="30"
    maxWaitMillis="10000"/>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- The contents of this file will be loaded for each web application -->
<Context>

    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <WatchedResource>WEB-INF/aas-web.xml</WatchedResource>
    <WatchedResource>${apusic.base}/conf/web.xml</WatchedResource>

    <!-- Uncomment this to disable session persistence across Aas restarts -->
    <!--
    <Manager pathname="" />
    -->

    <Resource name="jdbc/test"
                        auth="Container"
                        type="javax.sql.DataSource"
                        driverClassName="com.mysql.jdbc.Driver"
                        url="jdbc:mysql://127.0.0.1:3306/test_platform?useUnicode=true&amp;characterEncoding=utf-8"
                        username="root"
                        password="root"
                        maxTotal="100"
                        maxIdle="30"
                        maxWaitMillis="10000"/>

</Context>
```

可配置参数说明：

name：数据源 jndi 名称

auth：容器负责资源的连接

type：绑定类型

url：数据库连接 url

username：用户名

password：密码

maxTotal：最大连接数

maxIdle：最大空闲连接数

maxWaitMillis：获取连接最大等待时间

initialSize：初始化连接大小，默认 0

minIdle：最小空闲连接数

validationQuery：连接有效性检测语句

validationQueryTimeout：检测语句超时时间，默认不超时

testOnBorrow：true 或 false，从池中拿连接的时，是否检测连接有效性，默认 true

testOnReturn：true 或 false，还回连接到池里面时，是否检测连接有效性，默认 false

removeAbandonedOnBorrow：true 或 false，是否移除失效的连接，默认 false

logAbandoned：true 或 false，是否打印移除失效连接，默认 false

testWhileIdle：true 或 false，连接空闲时，是否检测连接有效性，默认 false

minEvictableIdleTimeMillis：连接空闲时间，超过该时间会从池中移除（单位：毫秒）

timeBetweenEvictionRunsMillis：每隔多少时间去检测一次空闲连接是否超时，默认不检测（单位：毫秒）

注意：驱动需要放入到 AMS 安装的 lib 目录下。

注意：在代码中要引用 jndi 需要加上前缀：java:comp/env，如上面的数据源的：java:comp/env/jdbc/test

10. 配置 HTTPS

10.1 单向认证--证书格式为 JKS

注意：配置了新的 endpoint 需要在 server 中进行引用，配置了线程池也需要在 server 中进行引用。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
        <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
        <SSLHostConfig>
            <Certificate
                certificateKeystoreFile="conf/cert/arsp_server_keystore.jks"
                certificateKeystoreType="JKS"
                certificateKeystorePassword="apusic" />
```

```
        </SSLHostConfig>

    </endpoint>
```

```
22      <executor name="http-thread-pool"
23          namePrefix="http-exec-"
24          maxThreads="200"
25          minSpareThreads="25"
26          prestartminSpareThreads="true"
27          maxQueueSize="5000"/>
28  </executors>
29
30  <endpoints>
31      <endpoint name="ams-http"
32          port="6888"
33          protocol="HTTP/1.1"
34          connectionTimeout="20000"
35          redirectPort="6443"
36          executor="http-thread-pool"
37          maxConnections="10000"
38          enableLookups="false"
39          acceptCount="100"
40          maxPostSize="104857600"
41          maxHttpHeaderSize="8192"
42          compression="on"
43          disableUploadTimeout="true"
44          compressionMinSize="2048"
45          compressibleMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf"
46          URIEncoding="utf-8"
47          processorCache="20000"
48          tcpNoDelay="true"
49          keepAliveTimeout="15000"
50          maxKeepAliveRequests="100">
51      </endpoint>
52
53      <endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
54          <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
55          <SSLHostConfig>
56              <Certificate
57                      certificateKeystoreFile="conf/cert/arsp_server_keystore.jks"
58                      certificateKeystoreType="JKS"
59                      certificateKeystorePassword="apusic" />
60          </SSLHostConfig>
61      </endpoint>
62  </endpoints>
63
64  <services>
65      <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
```

说明：

SSLEnabled：是否是 ssl 通讯

executor：使用的线程池

certificateKeystoreType：keystore 格式，JKS 或者 PKCS12

certificateKeystorePassword：keystore 密码

certificateKeystoreFile：keystore 文件

10.2 双向认证--证书格式为 JKS

注意:配置了新的 endpoint 需要在 server 中进行引用,配置了线程池也需要在 server 中进行引用。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
        <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
```

```
        <SSLHostConfig
                truststoreType="JKS"
                truststorePassword="apusic"
                truststoreFile="conf/cert/arsp_server_truststore.jks"
                certificateVerification="required" >
            <Certificate
                    certificateKeystoreFile="conf/cert/arsp_server_keystore.jks"
                    certificateKeystoreType="JKS"
                    certificateKeystorePassword="apusic" />
        </SSLHostConfig>
</endpoint>
```

```
    <endpoints>
        <endpoint name="ams-http"
            port="6888"
            protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="6443"
            executor="http-thread-pool"
            maxConnections="10000"
            enableLookups="false"
            acceptCount="100"
            maxPostSize="104857600"
            maxHttpHeaderSize="8192"
            compression="on"
            disableUploadTimeout="true"
            compressionMinSize="2048"
            compressibleMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf"
            URIEncoding="utf-8"
            processorCache="20000"
            tcpNoDelay="true"
            keepAliveTimeout="15000"
            maxKeepAliveRequests="100">
        </endpoint>

        <endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
            <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
            <SSLHostConfig
                    truststoreType="JKS"
                    truststorePassword="apusic"
                    truststoreFile="conf/cert/arsp_server_truststore.jks"
                    certificateVerification="required" >
                <Certificate
                        certificateKeystoreFile="conf/cert/arsp_server_keystore.jks"
                        certificateKeystoreType="JKS"
                        certificateKeystorePassword="apusic" />
            </SSLHostConfig>
        </endpoint>
    </endpoints>
```

说明：

SSLEnabled：是否是 ssl 通讯

executor：使用的线程池

truststoreType：信任库格式，JKS 或者 PKCS12

truststorePassword：信任库密码

truststoreFile：信任库文件

certificateVerification：是否需要验证客户端证书，即是否双向认证，默认不验证。

certificateKeystoreType：keystore 格式，JKS 或者 PKCS12

certificateKeystorePassword：keystore 密码

certificateKeystoreFile：keystore 文件

10.3 单向认证--证书格式为 PEM

注意：配置了新的 endpoint 需要在 server 中进行引用，配置了线程池也需要在 server 中进行引用。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
    <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
    <SSLHostConfig>
        <Certificate
            certificateKeyFile="conf/cert/private_key.pem"
            certificateKeyPassword="apusic"
            certificateFile="conf/cert/public_key.pem"
            certificateChainFile="conf/cert/chain.pem"
            type="RSA"/>
    </SSLHostConfig>
</endpoint>
```

```
68          -->
69
70 ⊟        <endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
71              <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
72 ⊟            <SSLHostConfig>
73 ⊟                <Certificate
74                          certificateKeyFile="conf/cert/private_key.pem"
75                          certificateKeyPassword="apusic"
76                          certificateFile="conf/cert/public_key.pem"
77                          certificateChainFile="conf/cert/chain.pem"
78                          type="RSA"/>
79              </SSLHostConfig>
80          </endpoint>
81
```

说明：

certificateKeyFile：私钥文件

certificateKeyPassword：私钥密码，可选

certificateFile：公钥文件

certificateChainFile：证书链文件，可选

10.4 双向认证--证书格式为 PEM

<endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">

<UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />

<SSLHostConfig

caCertificateFile="conf/cert/public_ca_key.pem"

certificateVerification="required">

<Certificate

certificateKeyFile="conf/cert/private_key.pem"

certificateKeyPassword="apusic"

certificateFile="conf/cert/public_key.pem"

certificateChainFile="conf/cert/chain.pem"

type="RSA"/>

</SSLHostConfig>

</endpoint>

```
69
70        <endpoint name="ams-https" port="8443" protocol="HTTP/1.1" SSLEnabled="true" executor="http-thread-pool">
71            <UpgradeProtocol className="com.apusic.connector.http2.Http2Protocol" />
72            <SSLHostConfig
73                caCertificateFile="conf/cert/public_ca_key.pem"
74                certificateVerification="required">
75                <Certificate
76                        certificateKeyFile="conf/cert/private_key.pem"
77                        certificateKeyPassword="apusic"
78                        certificateFile="conf/cert/public_key.pem"
79                        certificateChainFile="conf/cert/chain.pem"
80                        type="RSA"/>
81            </SSLHostConfig>
82        </endpoint>
83
```

说明：

certificateKeyFile：私钥文件

certificateKeyPassword：私钥密码，可选

certificateFile：公钥文件

certificateChainFile：证书链文件，可选

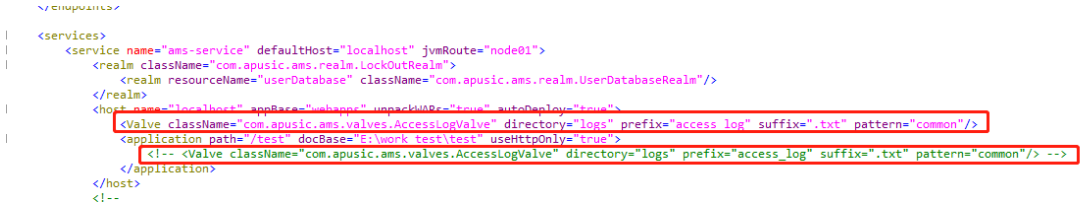
caCertificateFile：根证书文件，可选

certificateVerification：是否需要验证客户端证书，即是否双向认证，默认不验证。

11. 配置访问日志

访问日志可配置在 application 或者 service 里面。配置在 host 里面则该 host 下的应用访问都会被记录日志。

<Valve className="com.apusic.ams.valves.AccessLogValve" directory="logs" prefix="access_log" suffix=".txt" pattern="common"/>



日志详细配置：

directory：日志存储目录

prefix：日志文件前缀

suffix：日志文件后缀

fileDateFormat：文件时间格式，一般文件命为 prefix（前缀）+.+时间格式（一般是按天算）+.+suffix（后缀），如果想改为每小时一个文件可设置为 fileDateFormat="yyyy-MM-dd.HH"

pattern：参考下面内容,

%a：远程 IP 地址

%A：本地 IP 地址

%b：发送的字节数，不包括 HTTP 头，或“-”如果没有发送字节

%B：发送的字节数，不包括 HTTP 头

%h：远程主机名

%H：请求协议

%l (小写的 L)- 远程的用户名

%m：请求方法

%p：本地端口

%q：查询字符串

%r：第一行的要求

%s：响应的 HTTP 状态代码

%S：用户会话 ID

%t：日期和时间，在通用日志格式

%u：远程用户身份验证

%U：请求的 URL 路径

%v：本地服务器名

%D：处理请求的时间（以毫秒为单位）

%T：处理请求的时间（以秒为单位）

%I （大写的 i）：当前请求的线程名称

此外，可以指定以下别名来设置：

common：%h %l %u %t "%r" %s %b

combined：%h %l %u %t "%r" %s %b "%{Referer}i" "%{User-Agent}i"

另外，还可以将 request 请求的查询参数、session 会话变量值、cookie 值或 HTTP 请求/响应头内容的变量值等内容写入到日志文件。类似 apache 的语法：

%{XXX}i xxx 代表传入的头(HTTP Request)

%{XXX}o xxx 代表传出的响应头(Http Resonse)

%{XXX}c　xxx 代表特定的 Cookie 名

%{XXX}r　xxx 代表 ServletRequest 属性名

%{XXX}s　xxx 代表 HttpSession 中的属性名

12. 配置 Session 存储

AMS 单机方式有两种 session 存储方式：StandardManager（默认）、PersistentManager

StandardManager 方式把 session 放在内存中，停止时才把内存中的 session 保存到文件，再次启动时从文件读取。

PersistentManager 方式又分为文件或者数据库存储。

12.1 配置本地内存存储（StandardManager）

默认的 session 存储就是本地内存存储（StandardManager），该方式 session 是放在内存中，停止时才把内存中的 session 保存到文件，再次启动时从文件读取。

配置如下：

```
<Manager
    maxActiveSession="-1">
</Manager>
```

```
<services>
    <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
        <realm className="com.apusic.ams.realm.LockOutRealm">
            <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseReal
        </realm>
        <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
            <application path="/test" docBase="E:\work_test\test" useHttpOnly="true">
                <Manager
                    maxActiveSession="-1">
                </Manager>
            </application>
        </host>
```

参数说明：

maxActiveSession：最大会话数，-1 表示不限制，默认-1，超过后再创建 session 会直接抛异常。

12.2 配置本地文件存储

```
<services>
    <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
        <realm className="com.apusic.ams.realm.LockOutRealm">
            <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
        </realm>
        <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
            <application path="/test" docBase="F:\work_test\test" useHttpOnly="true">
                <Manager className="com.apusic.ams.session.PersistentManager"
                    saveOnRestart="true"
                    maxActiveSessions="1"
                    minIdleSwap="-1"
                    maxIdleSwap="-1"
                    maxIdleBackup="60">
                    <Store className="com.apusic.ams.session.FileStore"/>
                </Manager>
            </application>
        </host>
```

saveOnRestart：(true/false）配置服务重启对 session 的处理，若为 true，则关闭前把有效的 session 保存，启动后重新载入。

maxActiveSessions：活动状态 Session 的最大数，为-1 时则不限制，否则 Session Manager 将会把超出的 Session 对象转移到 Session Store 中。

minIdleSwap：Session 不活动的最短时间，超过该时间，Session Manager 可能会把该 Session 对象转移到 Session Store 中，单位为秒，该 Session 将不在内存中，如果配置了 minIdleSwap 则两个条件都需要满足。

maxIdleSwap：Session 不活动的最长时间，超过该时间，Session Manager 将会把该 Session 对象转移到 Session Store 中，该 Session 将不在内存中。如果配置了 minIdleSwap 则两个条件都需要满足。

maxIdleBackup： Session 不活动的最长时间，超过该时间，Session Manager 将会把该 Session 对象备份到 Session Store 中，但该 Session 对象依然存在内存中。

12.3 配置数据库存储

<Manager className="com.apusic.ams.session.PersistentManager"

saveOnRestart="true"

maxActiveSessions="1"

minIdleSwap="-1"

maxIdleSwap="-1"

maxIdleBackup="60">

<Store

className="com.apusic.ams.session.JDBCStore"

driverName="com.mysql.jdbc.Driver"

connectionURL="jdbc:mysql://127.0.0.1:3306/test_platform?useUnicode=true&characterEncoding=utf-8"

connectionName="root"

connectionPassword="root"

sessionTable="ams_session"

sessionIdCol="session_id"

sessionDataCol="session_data"

sessionValidCol="session_valid"

sessionMaxInactiveCol="max_inactive"

sessionLastAccessedCol="last_access"

sessionAppCol="app_name"/>

</Manager>

```
    <services>
        <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
            <realm className="com.apusic.ams.realm.LockOutRealm">
                <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
            </realm>
            <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
                <application path="/test" docBase="E:\work test\test" useHttpOnly="true">
                    <Manager className="com.apusic.ams.session.PersistentManager"
                        saveOnRestart="true"
                        maxActiveSessions="1"
                        minIdleSwap="-1"
                        maxIdleSwap="-1"
                        maxIdleBackup="60">
                        <Store
                            className="com.apusic.ams.session.JDBCStore"
                            driverName="com.mysql.jdbc.Driver"
                            connectionURL="jdbc:mysql://127.0.0.1:3306/test_platform?useUnicode=true&amp;characterEncoding=utf-8"
                            connectionName="root"
                            connectionPassword="root"
                            sessionTable="ams_session"
                            sessionIdCol="session_id"
                            sessionDataCol="session_data"
                            sessionValidCol="session_valid"
                            sessionMaxInactiveCol="max_inactive"
                            sessionLastAccessedCol="last_access"
                            sessionAppCol="app_name"/>
                    </Manager>
                </application>
            </host>
            <!--
```

注意：需要自己创建表。

13. 集群方案

13.1 方案一：Session 同步

Session 同步需要满足以下几个要求：

（1） Session 的属性必须实现 java.io.Serializable 接口
（2） 负载均衡器需要配置会话粘滞模式（sticky session mode），测试时可不开启会话粘滞
（3） 多台机器时间需要同步
（4） 保证多播端口可用
（5） 应用的 web.xml 需要加入 <distributable/>
（6） 应用使用文件夹方式部署

13.1.1 添加集群配置

在需要加入集群的机器中的 apusic.conf 文件的 host 或者 service 里面添加如下内容（注意：在 service 下添加表示对所有虚拟主机均启用集群功能，否则对某个虚拟主机启用集群功能）

```
<Cluster  className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
                    channelSendOptions="6">
                  <Manager  className="com.apusic.ams.ha.session.BackupManager"
                     expireSessionsOnShutdown="false"
                     notifyListenersOnReplication="true"/>
                  <Channel  className="com.apusic.ams.tribes.group.GroupChannel">
                       <Membership  className="com.apusic.ams.tribes.membership.Mc
astService"
                                address="228.0.0.4"
                                port="45564"
                                frequency="500"
                                dropTime="3000"/>
                       <Receiver  className="com.apusic.ams.tribes.transport.nio.NioRec
eiver"
                                address="172.20.50.43"
                                port="5000"
```

```
                    selectorTimeout="100"
                    maxThreads="6"/>

                <Sender  className="com.apusic.ams.tribes.transport.ReplicationT
ransmitter">
                    <Transport  className="com.apusic.ams.tribes.transport.nio.P
ooledParallelSender"/>
                </Sender>
                <Interceptor  className="com.apusic.ams.tribes.group.interceptor
s.TcpFailureDetector"/>
                <Interceptor  className="com.apusic.ams.tribes.group.interceptor
s.MessageDispatch15Interceptor"/>
                <Interceptor  className="com.apusic.ams.tribes.group.interceptor
s.ThroughputInterceptor"/>
            </Channel>
            <Valve  className="com.apusic.ams.ha.tcp.ReplicationValve"  filter=""
/>
            <Valve  className="com.apusic.ams.ha.session.JvmRouteBinderValve"
/>

            <ClusterListener  className="com.apusic.ams.ha.session.ClusterSession
Listener"/>
        </Cluster>
```

```
<services>
    <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
        <realm className="com.apusic.ams.realm.LockOutRealm">
            <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
        </realm>
        <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
            <application path="/test" docBase="E:\work_test\test" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext"/>
        </host>
        <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
             channelSendOptions="6">
            <Manager className="com.apusic.ams.ha.session.DeltaManager"
               expireSessionsOnShutdown="false"
               notifyListenersOnReplication="true"/>
            <Channel className="com.apusic.ams.tribes.group.GroupChannel">
                <Membership className="com.apusic.ams.tribes.membership.McastService"
                    address="228.0.0.4"
                    port="45564"
                    frequency="500"
                    dropTime="3000"/>
                <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
                  address="auto"
                  port="5000"
                  selectorTimeout="100"
                  maxThreads="6"/>
                <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
                    <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
                </Sender>
                <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
                <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
                <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
            </Channel>
            <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
            <Valve className="com.apusic.ams.ha.session.JvmRouteBinderValve"/>
            <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
        </Cluster>
    </service>
</services>

<servers>
    <server name="ams-server" services="ams-service" endpoints="ams-http" executors="http-thread-pool"/>
</servers>
```

注意：需要把 NioReceiver 里面的 address 修改为当前机器的 ip 地址。

重要配置详解：

Cluster：

className：实现集群功能的类名称

channelSendOptions：消息发送选项，可以设置为 2（确认发送），4（同步发送），8（异步发送）以及它们的逻辑或运算结果，默认是 8，如果希望异步发送且确认发送结果，那么设置为 2+8=10。

Manager：

className ：集群会话管理器类，会话改变了会同步给集群中的其它节点。

expireSessionsOnShutdown：设置为 true 时，一个节点关闭，将导致集群下的所有 Session 失效。

notifyListenersOnReplication：如果希望当集群中的节点上复制或删除会话属性时通知会话监听器就设置 notifyListenersOnReplication 为 true。

mapSendOptions ：为 6 表示 BackupManager 同步发送消息，参考 <Cluster> 的 channelSendOptions。

Membership：

className ：集群节点信息管理类名

address：多播地址

port：多播端口

frequency：发送心跳时间间隔，单位毫秒

dropTime：该 dropTime 配置的时间内没有收到某个心跳，那么这个心跳所属节点就会从当前节点维护的节点列表删除，单位毫秒

Receiver：接收器，负责接收消息

className ：消息接收类名

address：节点监听地址

port：节点监听端口

selectorTimeout：接收器的 selector 超时时间

maxThreads：最大线程数

Sender：发送器，负责将消息发送给其他节点的 Reciver

Interceptor：拦截器

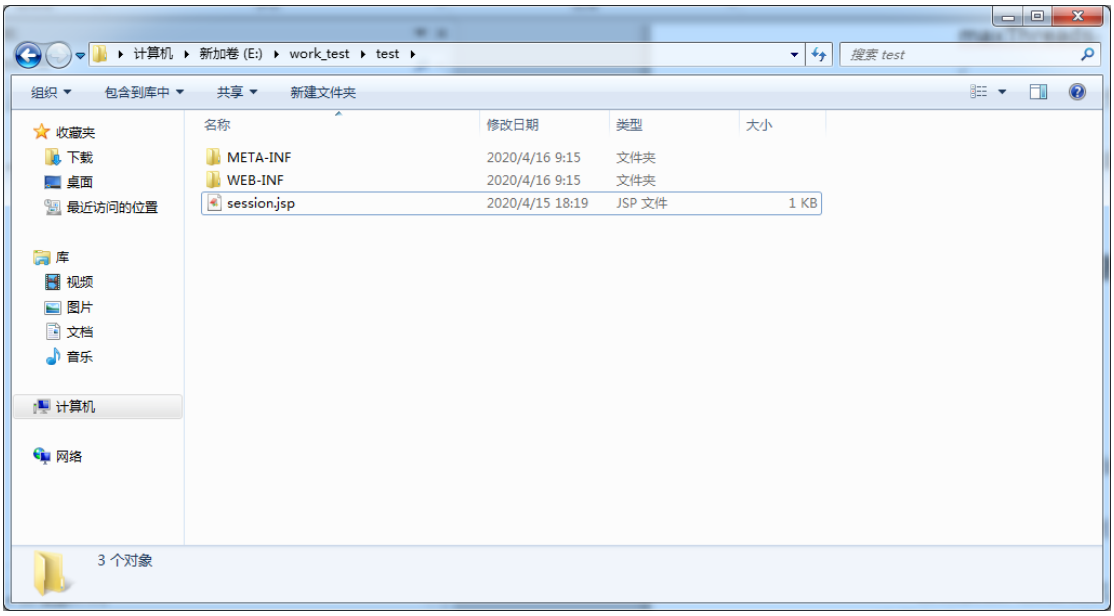
13.1.2 部署应用和设置 jvmRoute

（1）测试应用

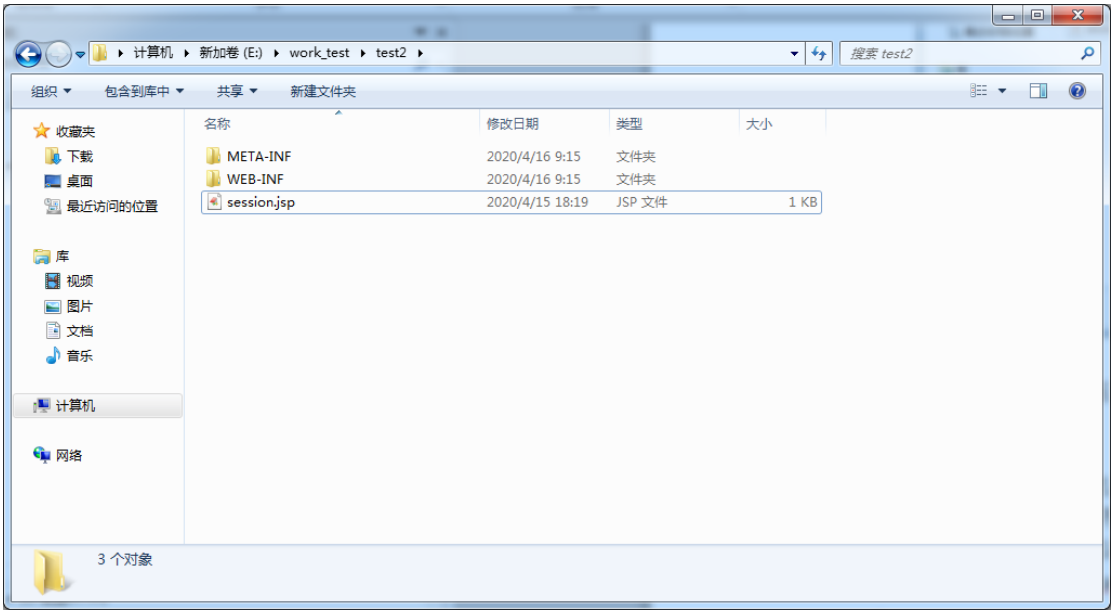
test.war

test.war 是我用于测试集群的应用。里面只有一个 session.jsp，需要把应用解压进行部署。

我这里节点 1 是解压到 e:/work_test/test



我这里节点 2 是解压到 e:/work_test/test2



session.jsp 内容：

```
<%@ page import="java.text.SimpleDateFormat" %>
<%@ page import="java.util.Date" %>
<%@ page import="java.util.logging.Logger" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    Logger logger = Logger.getLogger("test");
    String str = (String) request.getSession().getAttribute("test");
    if(null == str) {
        str = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
        request.getSession().setAttribute("test", str);
        logger.info("session put: " + str);
    } else {
        logger.info("session get: " + str);
    }
%>
```

往 session 里面获取数据，没有数据再往里放数据。

修改 web.xml 文件，添加<distributable/>

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlr
  <display-name>test</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <distributable/>

</web-app>
```

（2）部署应用和修改配置

修改 apusic.conf 文件的 service 加上 jvmRoute="nodeXXX".
确保每个 jvmRoute 都是不同。还需要在 application 加上 className="com.apusic.ams.ha.context.ReplicatedContext"。

我这里只有两个节点，分别为 node01、node02

```
<services>
    <service name="ams-service" defaultHost="localhost" jvmRoute="node02">
        <realm className="com.apusic.ams.realm.LockOutRealm">
            <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
        </realm>
        <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
            <application path="/test" docBase="E:\work_test\test2" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext"/>
        </host>
        <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
             channelSendOptions="6">

      <Manager className="com.apusic.ams.ha.session.DeltaManager"
               expireSessionsOnShutdown="false"
               notifyListenersOnReplication="true"/>

      <Channel className="com.apusic.ams.tribes.group.GroupChannel">
        <Membership className="com.apusic.ams.tribes.membership.McastService"
                    address="228.0.0.4"
                    port="45564"
                    frequency="500"
                    dropTime="3000"/>
        <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
                  address="auto"
                  port="5000"
                  selectorTimeout="100"
                  maxThreads="6"/>

        <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
          <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
        </Sender>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
      </Channel>

      <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
      <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
    </Cluster>
    </service>
</services>
```

```
</endpoints>

<services>
    <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
        <realm className="com.apusic.ams.realm.LockOutRealm">
            <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
        </realm>
        <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
            <application path="/test" docBase="E:\work_test\test" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext"/>
        </host>
        <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
             channelSendOptions="6">
            <Manager className="com.apusic.ams.ha.session.DeltaManager"
              expireSessionsOnShutdown="false"
              notifyListenersOnReplication="true"/>
            <Channel className="com.apusic.ams.tribes.group.GroupChannel">
                <Membership className="com.apusic.ams.tribes.membership.McastService"
                    address="228.0.0.4"
                    port="45564"
                    frequency="500"
                    dropTime="3000"/>
                <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
                  address="auto"
                  port="5000"
                  selectorTimeout="100"
                  maxThreads="6"/>
                <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
                    <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
                </Sender>
                <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
                <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
                <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
            </Channel>
            <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
            <Valve className="com.apusic.ams.ha.session.JvmRouteBinderValve"/>
            <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
        </Cluster>
    </service>
</services>
```

13.1.3 配置 apache

安装好 apache 后，向配置文件 httpd.conf 末尾添加如下内容：

Listen 172.20.50.43:8080

ProxyPass / balancer://testcluster/ stickysession=JSESSIONID

ProxyPassReverse / balancer://testcluster

<Proxy balancer://testcluster>

BalancerMember http://172.20.50.43:6888/ route=node01

BalancerMember http://172.20.50.43:8888/ route=node02

</Proxy>

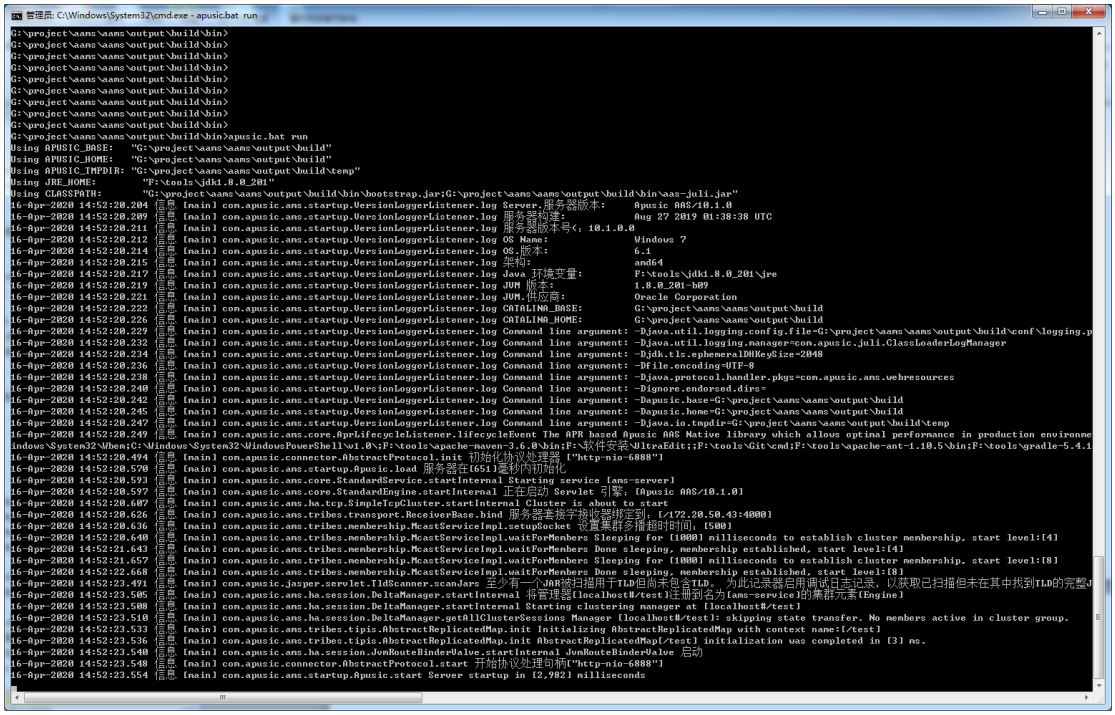
注意：上面的 172.20.50.43:8080 是我本地的 apache 监听的 ip 地址，需要修改为自己的地址。

172.20.50.43:6888、172.20.50.43:8888 分别是我的两个节点，route=xxx 分别填写上面 service 中的 jvmRoute 的值。

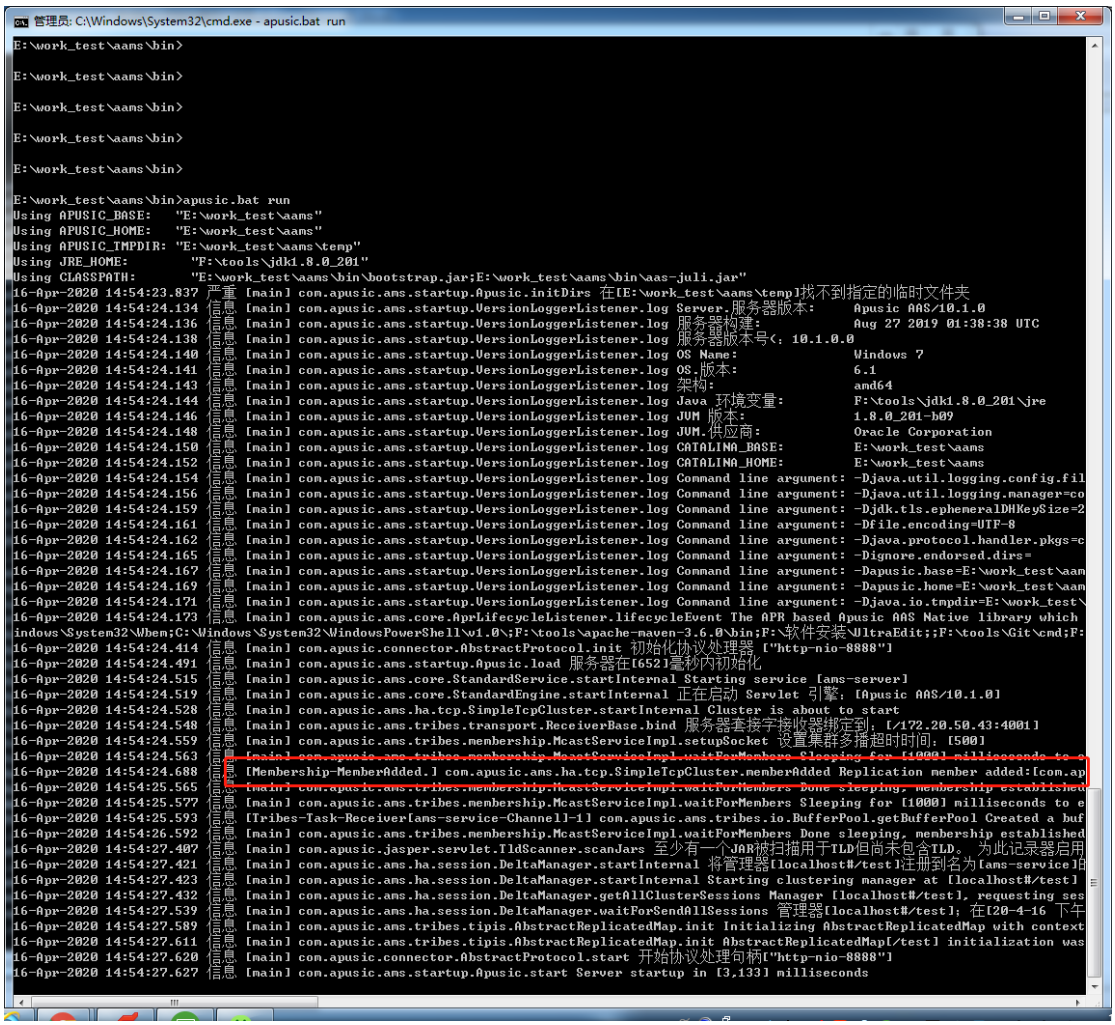
stickysession=JSESSIONID：会话粘滞，测试时，可先不配置为会话粘滞。

13.1.4 验证集群

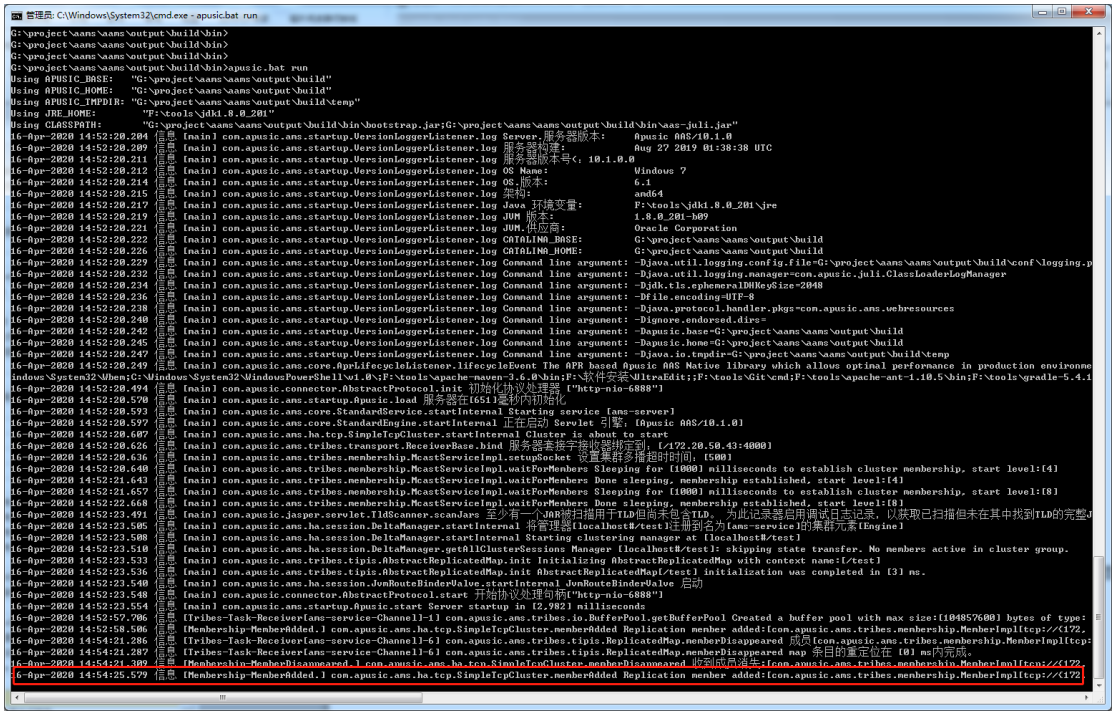
启动节点 node01：



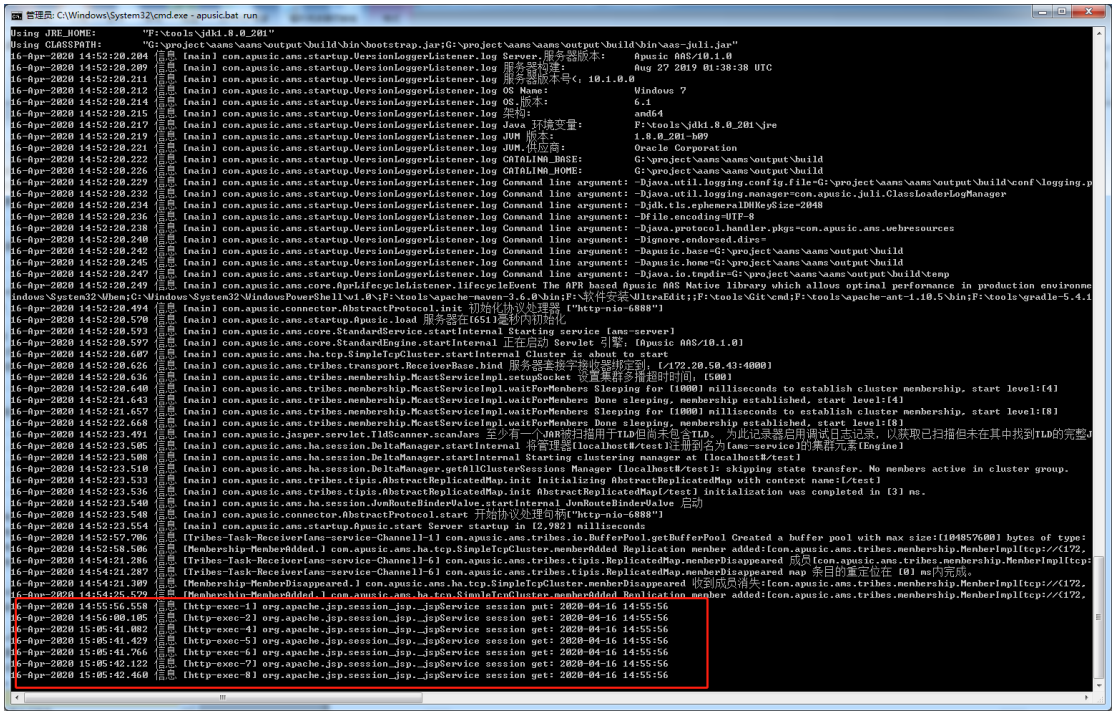
启动节点 node02：

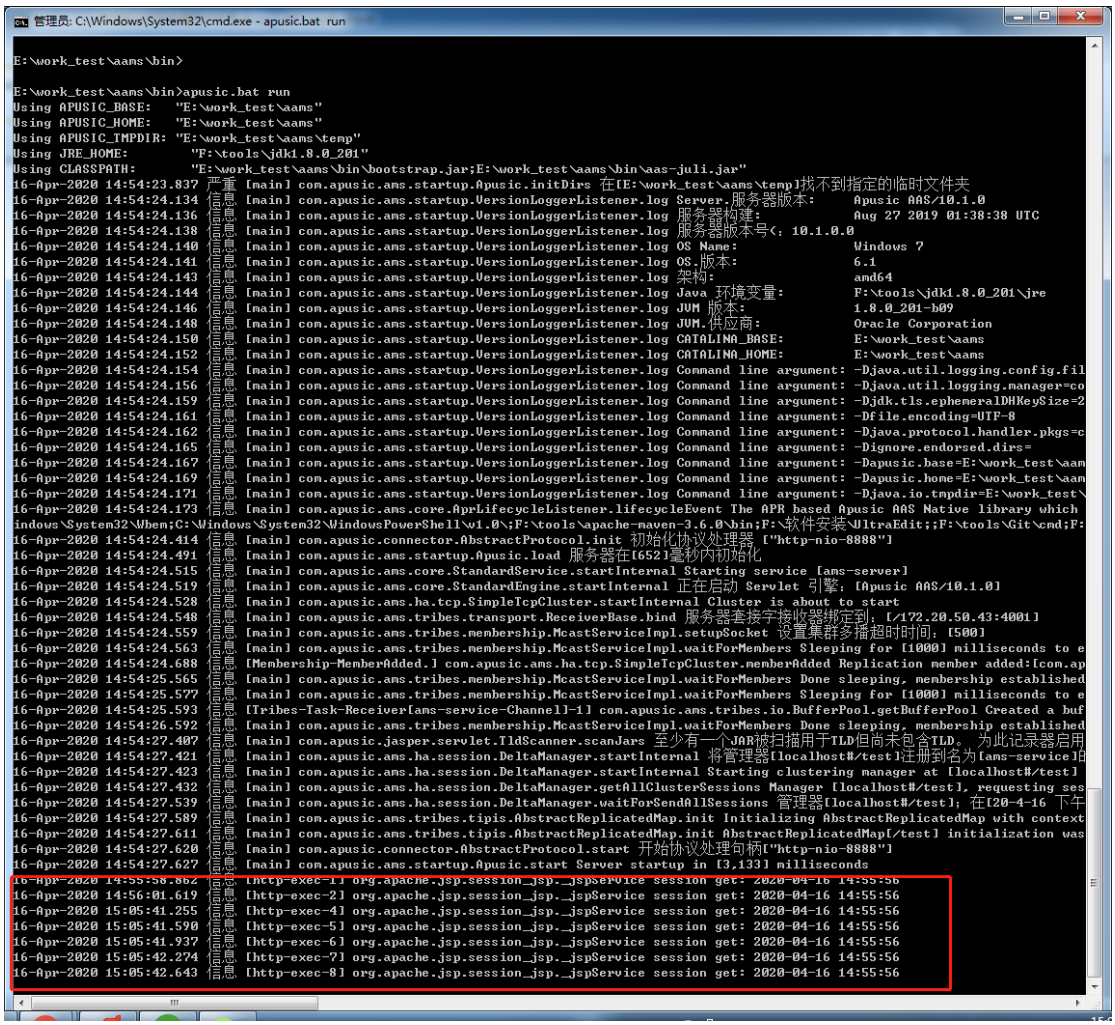


节点 node01 输出节点加入信息：



访问 http://172.20.50.43:8080/test/session.jsp，多刷新几次：





13.2 方案二：Session 集中式存储

后续版本提供支持。

天燕云公众号

徐少春个人号

Kingdee 天燕云

金蝶天燕云计算股份有限公司

国家规划布局内重点软件企业

金蝶集团旗下政府云服务企业

中电科集团太极股份成员企业

www.APUSIC.com

4008-555-800