Apusic 金蝶天燕

生成时间：2026-05-19

安装手册

金蝶Apusic统一管理平台V2.3

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 产品介绍

金蝶Apusic统一管理平台（Apusic Unified Management Platform ,以下简称"AUMP"）是金蝶天燕云计算股份有限公司经过多年经验积累，维护实践、自主研发和技术创新的统一中间件管理平台产品。

AUMP从统一的用户视角出发，对全系列中间件产品进行接入，实现对中间件的统一管理、统一运维、统一监控。保障IT基础设施的高可用和业务系统正常稳定可靠运行，极大提高信息中心IT运维的效率，使得对IT基础架构管理从被动分散的维护转变为主动集中的控制和自动化，智能化的管理。

2 范围和读者

本手册介绍AUMP产品的使用详细说明，适用于AUMP产品的用户，AUMP产品技术顾问，AUMP产品维护人员，以及希望学习了解AUMP产品的相关人员。

2.1 约定与术语

一些约定的缩略词诠释

AUMP 金蝶Apusic统一管理平台（Apusic Unified Management Platform）

AAS 金蝶Apusic应用服务器（Apusic Application Server）

3 安装环境要求

3.1 配置要求

安装AUMP产品的最低配置要求见下表

资源环境	要求
操作系统	Linux Red Hat 5.2或以上(及其他Kernel 2.25或以上linux版本)
CPU	Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz或以上
MySQL	5.6或以上
Redis	3.0或以上
内存	8G或以上
硬盘	可用空间128G或以上
浏览器	FireFox 21及以上、Chrome 23及以上、IE 10及以上

3.2 推荐配置

安装AUMP产品的推荐的配置见下表：

管理节点：

资源环境	要求
操作系统	Linux Red Hat 5.2或以上(及其他Kernel 2.25或以上linux版本) /银河麒麟
CPU	Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz或以上/ 鲲鹏8核
MySQL	5.6或以上
Redis	5.0或以上
内存	16G或以上
硬盘	可用空间500G或以上
浏览器	FireFox 21及以上、Chrome 60及以上

控制节点：

资源环境	要求
操作系统	Linux Red Hat 5.2或以上(及其他Kernel 2.25或以上linux版本) / 银河麒麟
CPU	Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz或以上 / 鲲鹏16核
内存	32G或以上
硬盘	可用空间500G或以上

4 安装概述

4.1 介质说明

AUMP 产品包括如下文件，请在安装前逐一检查。不同平台请使用对应的产品安装包，若产品介质名称中不包含平台架构的字样，则适用于所有平台部署。

以下以x86_64的产品包介质为例说明:

模块	文件名	说明
AUMP管理平台	auc-v2.0.tar.gz	统一web控制台应用
中间件统一管理服务	aump-admin-2.0.zip	
webshell控制台服务	web-shell-1.0.1.zip	
代理执行器	executor-agent.zip	
minio安装包	minio	
AUMP控制器	ansible.zip	控制器ansible配置
控制器集成的软件包，如AAS v9,ADMQ等	packagefiles.zip	
AMP监控服务	请参考附件《AMP安装部署》	请参考附件《AMP安装部署》

4.2 配置说明

AUMP管理平台基于springboot框架开发，配置遵循springboot框架规范。为方便安装，平台内置了多种数据库配置。

文件名	说明
application.yml	基本配置，可通过spring.profiles.active指定具体配置
application-mysql.yml	MySQL数据库作为数据持久化存储的配置，默认使用用该文件
application-dm.yml	达梦数据库作为数据持久化存储的配置
application-kingbasees.yml	人大进仓KingbaseES作为数据持久化存储的配置

5 安装说明

5.1 安装控制器

AUMP控制器用于批量控制中间件所在服务器，多数据中心场景需在每个数据中心部署。

5.1.1 安装准备

- 安装ansible

```
yum install -y ansible
unzip ansible.zip -d /etc
y
```

修改/etc/ansible/ansible.cfg（为了防止日志过大问题）,这一行注释掉

```
# logging is off by default unless this path is defined
# if so defined, consider logrotate
#log_path = /var/log/ansible.log
```

- 安装软件包

```
mkdir -p /opt/aump
unzip packagefiles.zip -d /opt/aump
```

6 单机部署

- **安装JDK**，运行需要JDK8 环境，参考附录JDK 安装说明。
- **安装缓存服务**：运行需要缓存服务，默认使用Redis作为缓存服务，参考附录 Redis 安装说明，可使用AMDC代替。
- **安装数据库**：运行需要数据库服务，默认使用MySQL，参考附录 MySQL安装说明。可使用达梦7，金仓V8代替。

6.0.1 安装AUC

1.解压

```
tar -zxvf auc-v2.0.tar.gz -C /opt/aump
```

2.初始化数据库

```
create database auc;
use auc;
source /opt/aump/auc/sql/mysql/create.sql;
source /opt/aump/auc/sql/mysql/initial.sql;
```

3.修改配置

根据实际的数据库选择对应配置，本文以mysql为例，修改conf/application.yml

```
spring:
  profiles:
    active: mysql
```

配置数据库连接、redis连接等，修改config/application-mysql.yml

```
  datasource:
    type: com.zaxxer.hikari.HikariDataSource
    url: jdbc:mysql://localhost:3306/auc?
useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC
    username: root
    password: root
```

```
  redis:
    timeout: 3600
    host: localhost
    port: 6379
```

4.启动

```
nohup /opt/aump/auc/bin/startup.sh &
```

5.验证

浏览器访问 http://localhost:9000

账号 admin 密码 Admin$123456

6.0.2 安装AMP

请参考AMP产品安装手册文档。

6.0.3 安装Webshell

1.解压

```
unzip web-shell-1.0.1.zip -d /opt/aump
```

2.启动

```
cd /opt/aump/web-shell-1.0.1/
nohup ./startup.sh > ./webshell.log 2>&1 &
```

3.验证

浏览器访问 http://localhost:9999

6.0.4 安装aump-admin

1.解压安装包

```
unzip aump-admin-2.0.zip -d /opt/aump
```

2.初始化数据库

```
create database aump;
use aump;
source /opt/aump/aump-admin/sql/mysql/aump_admin.sql;
```

3 executor-agent代理执行器安装

3.1启动代理执行器服务程序

进入安装目录

```
./start.sh
```

启动执行器代理程序，默认使用8088端口

指定其他端口启动执行器程序，如8888端口，修改start.sh， 修改启动端口

```
PORT=8888
```

3.2 停止代理执行器服务程序

进入安装目录

```
./stop.sh
```

3.3 附录

代理执行器更多参数说明

直接执行代理程序./executor-agent

```
[root@linux-4-121 executor-agent]# ./executor-agent --help
Usage of ./executor-agent:
  -help
```

```
        show context-sensitive help info
  -listen string
        server listen address (default "0.0.0.0:8088")
  -version
        show server version info
```

3.4指定端口启动

不使用start.sh启动脚本，直接指定可执行程序端口方式启动，如8088端口

```
./executor-agent --listen="0.0.0.0:8088"
```

4.多数据中心配置

数据中心 安装下列顺序创建sql

id,描述,ip范围,位置,数据中心名称,数据中心类型,由谁创建，创建日期，最后更新人，最后更新日期，提供类型，状态 （默认数据库有1条内置数据中心，以下为实例）

```
INSERT INTO `data_center` VALUES ('1', null, '1.2.3.4-128', 'xxx', '内置数据中心', null, null,
'2022-12-13 13:52:24', null, '2022-12-13 13:52:28', null, '0');
```

ansible执行器主机 安装下列顺序创建sql

id,创建人,创建时间,最后更新人,最后更新日期,数据中心id（数据中心表查询相关id）,安装了ansible的主机（并有相关的ansible脚本）和executor-agent代理执行器,端口号,密码,用户名

```
INSERT INTO `excutor_host` VALUES ('1', 'admin', '2022-11-10 14:54:57', 'admin', '2022-11-10
14:55:04', '1', '172.24.4.138', '8088', 'Apusic1qazXSW@', 'root');

INSERT INTO `excutor_host` VALUES ('2', 'admin', '2022-11-20 14:54:05', 'admin', '2022-11-20
14:54:05', '1', '172.24.4.63', '8088', 'Apusic1qazXSW@', 'root');
```

执行excutor_host表，填入ansible的相关的信息，

5.minio安装

单机安装部署 单 minio 下面说明linux环境安装单机版本的minio服务 下载相应架构的minio二进制产品包 将二进制产品包上传到服务器，这里是上传到了/home/minio目录下。 编写启动minio服务脚本start.sh

```
#!/bin/bash
export MINIO_ACCESS_KEY=minioadmin
export MINIO_SECRET_KEY=minioadmin
nohup ./minio server --address ":9010" --console-address ":9001" /home/minio/data >
/home/minio/minio.log 2>&1 &
```

MINIO_ACCESS_KEY指定的是minio的登录账户 MINIO_SECRET_KEY指定的是minio的登录密码 --address指定的是minio服务端口 --console-address指定的是minio控制台服务端口 不指定上面两个参数时，minio上述两个端口默认使用9000 给minio执行文件，start.sh赋予可执行权限 chmod +x minio start.sh 编写停止minio服务stop.sh脚本

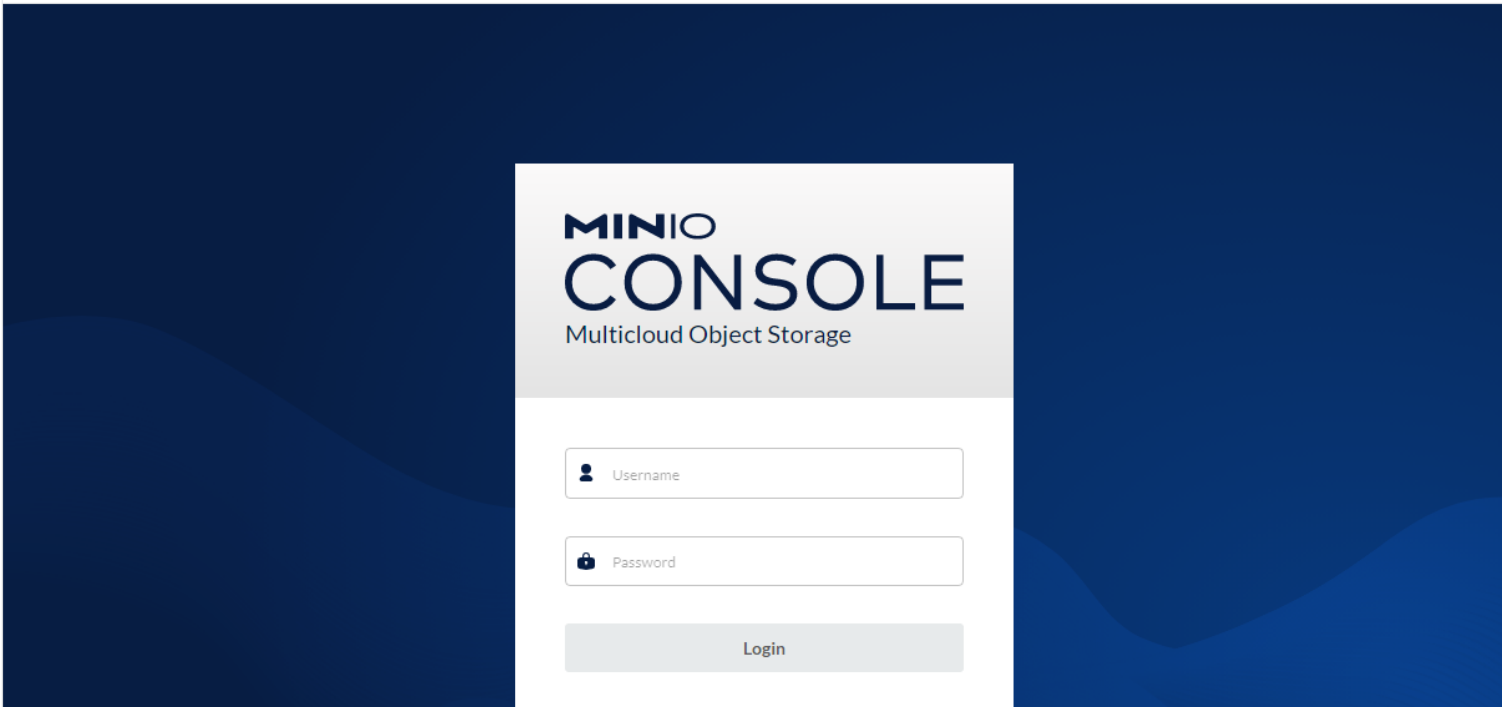
```
#!/bin/bash
PID=$(ps -ef | grep minio | grep -v grep | awk '{ print $2 }')
if [ -z "$PID" ]
then
```

```
echo minio is already stopped!
else
echo kill $PID
kill $PID
echo stop minio success!
fi
```

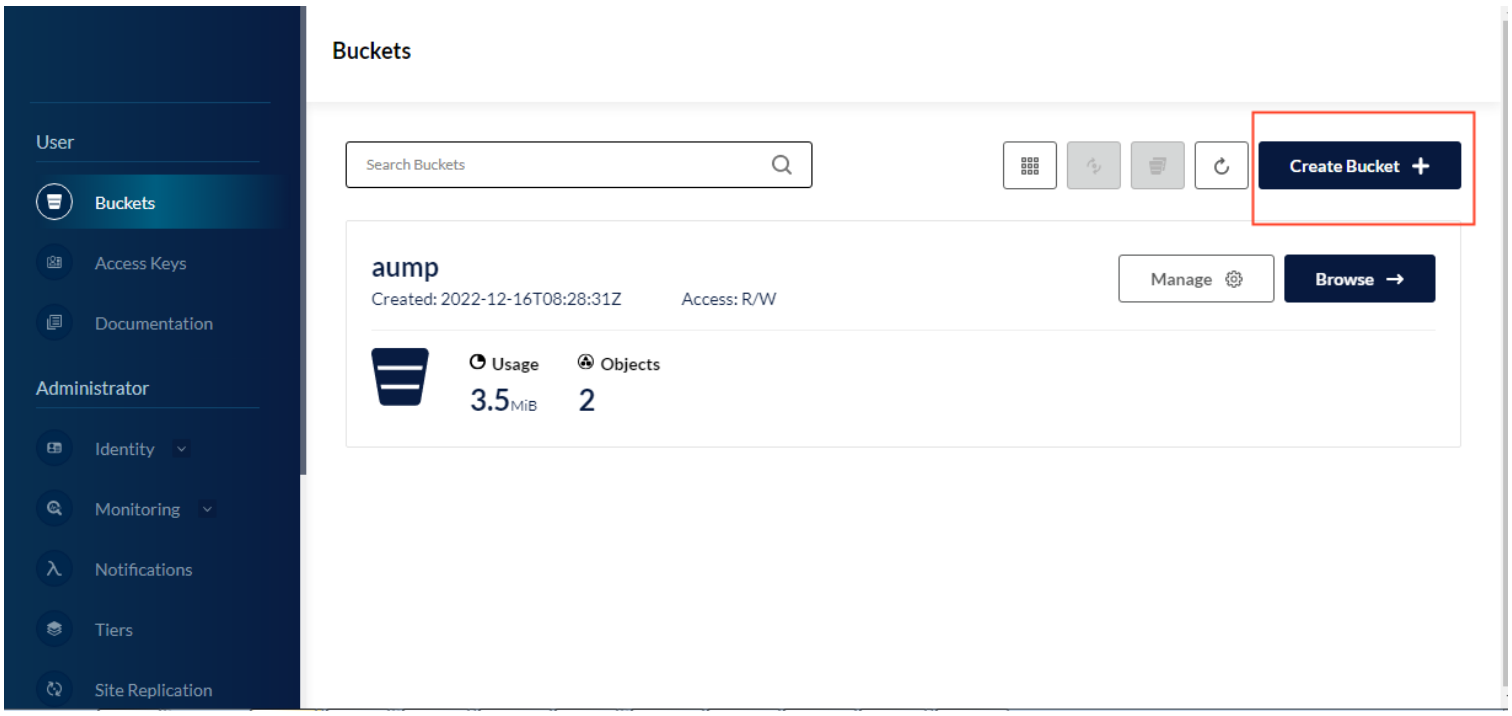
给stop.sh赋予可执行权限 chmod +x stop.sh 启动启 minio服务服 执行./start.sh启动minio服务。 停止停 minio服务服 执行./stop.sh停止minio服务。

minio配置

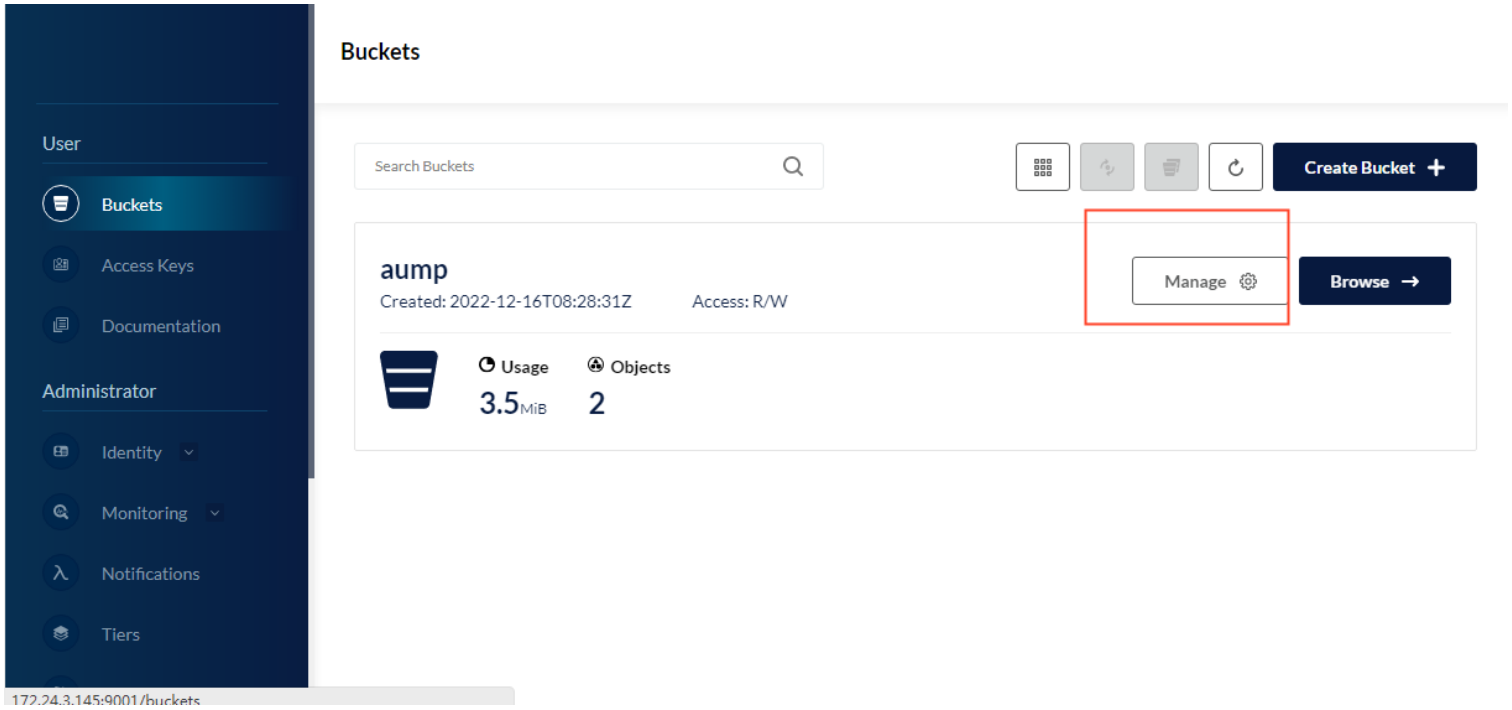
输入minio地址 http://172.24.3.145:9001/login(配置的地址)账号密码为minioadmin

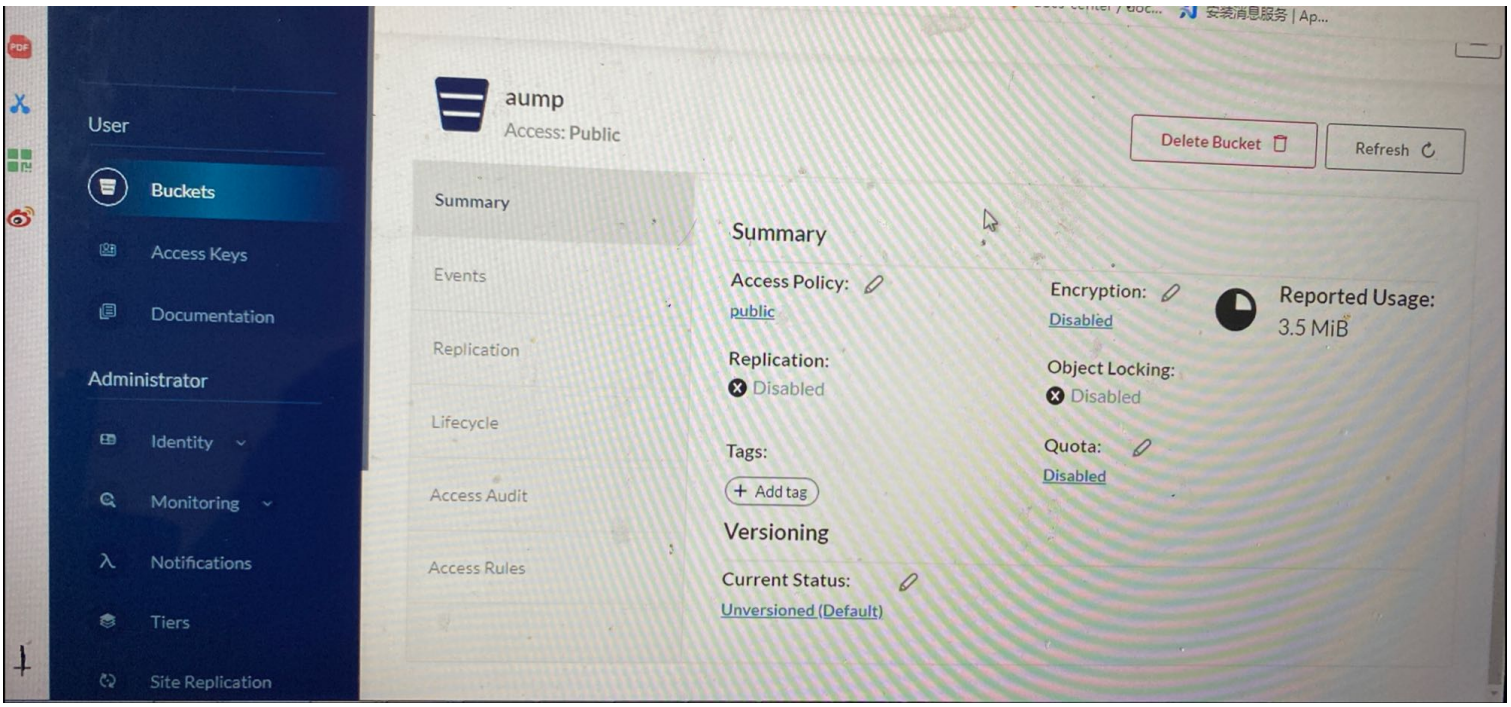


创建相关的bucket名字为aump

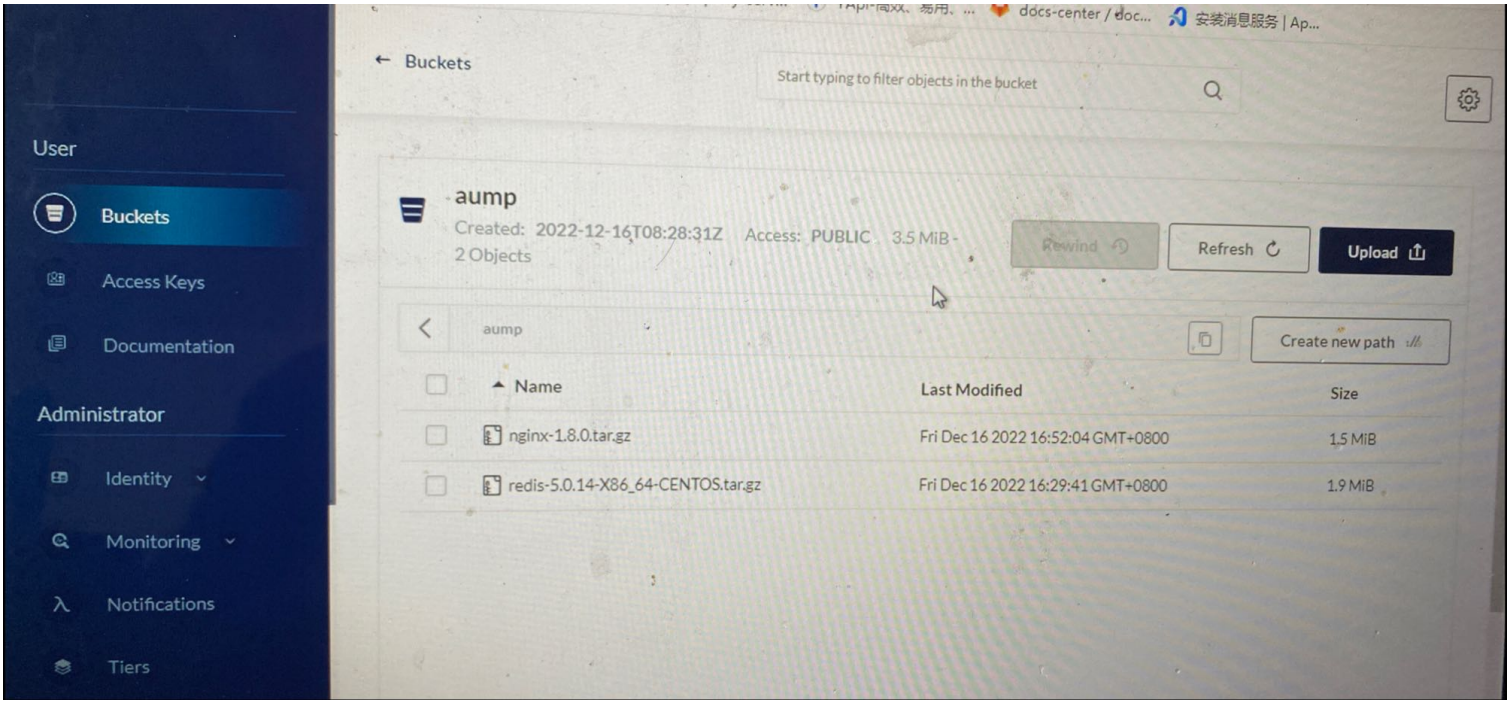


点击manager按钮将access police 改为public

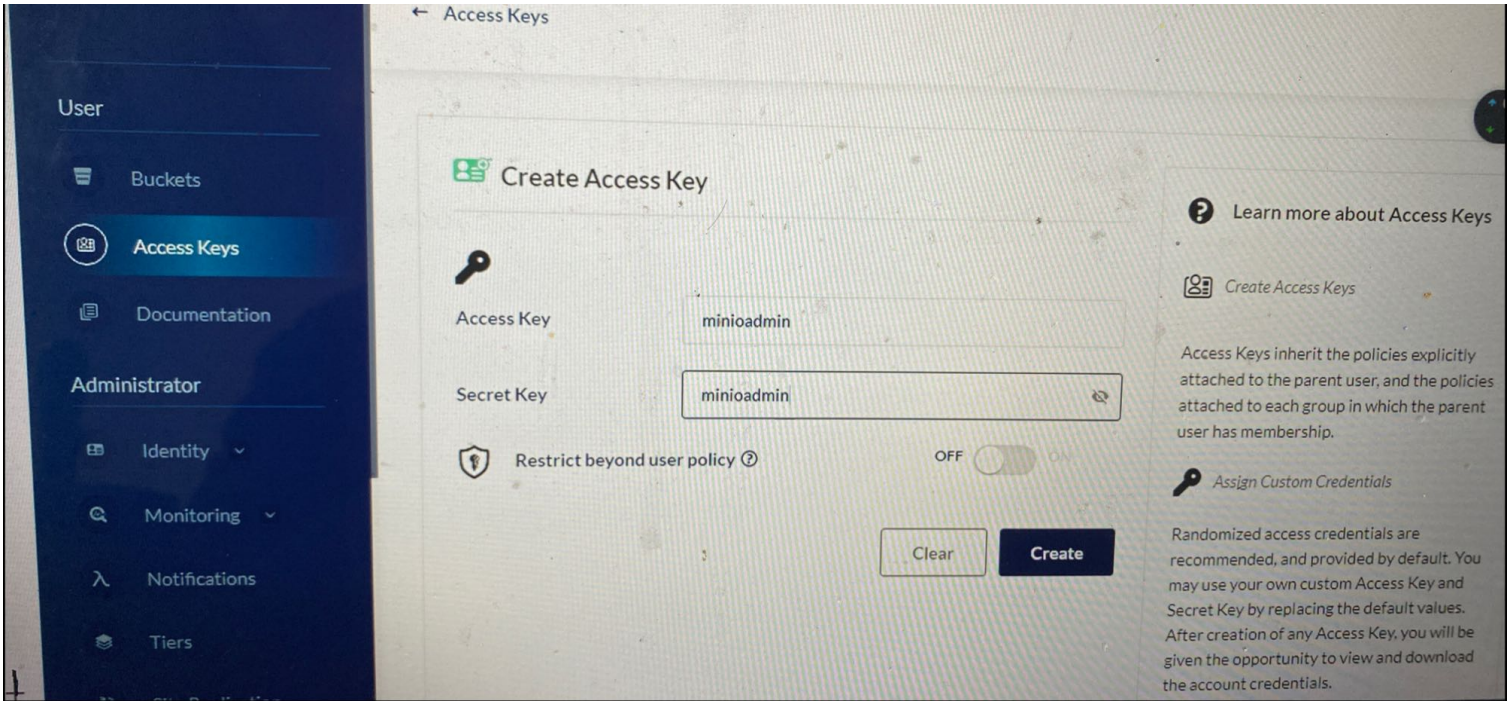




点browse将安装介质的软件包files下的包上传上去



点击access keys创建key， accesskey为aumpminioadmin，secretkey为aumpminioadmin



6.修改配置

根据实际的数据库选择对应配置，本文以mysql为例，修改config/application.yml

```
spring:
  profiles:
    active: mysql
```

配置数据库连接、redis连接等，修改conf/application-mysql.yml

数据库连接配置:

```
  datasource:
    type: com.zaxxer.hikari.HikariDataSource
    url: jdbc:mysql://localhost:3306/aump-admin?
useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC
    username: root
    password: root
```

redis连接配置:

```
redis:
    timeout: 3600
    host: localhost
    port: 6379
```

auc与amp配置:

```
aump:
  console:
    # auc连接
    url: http://172.24.3.142:9000
    serv: aump
```

```
  mon:
    # amp连接
    authUrl: http://172.24.3.142:9002/ims-springboard-page.html
    url: http://172.24.3.142:9002
    domain: http://172.24.3.142:9002
```

控制器配置:

```
ansible:
  # （不用修改）
  hostpath: /etc/ansible/
  install: /etc/ansible/playbooks
  # 控制器ip(不用修改)
  hostaddr: 172.24.3.142
  # 控制器ssh用户(不用修改)
  username: root
  # （不用修改）
  deploypath: /usr/local
  # 控制器ssh密码(不用修改)
  pwd: a81dIbFQq6zi
  # （不用修改）
  port: 22
  # （不用修改）
  tmpdir: /tmp/
  # （不用修改）
  packagepath: /opt/aump/files/
```

软件库配置:

```
package:
 # 软件库用户
  user: minioadmin（相关的accesskey）
  # 软件库密码
  password: minioadmin（相关的secretkey）
  # 软件库url
  url: http://172.24.4.165:9010(根据实际配置的minio修改)
  # 软件库路径
  packagepath: aump(minio添加的bucket文件)
  # 软件库类型
  type: minio(内置的相关定义不需要修改)
```

webshell配置:

```
webshell:
  url: http://172.24.3.142:9999
```

7.生成密钥

```
ssh-keygen -t rsa -P "" -f /root/.ssh/id_rsa
```

8.启动

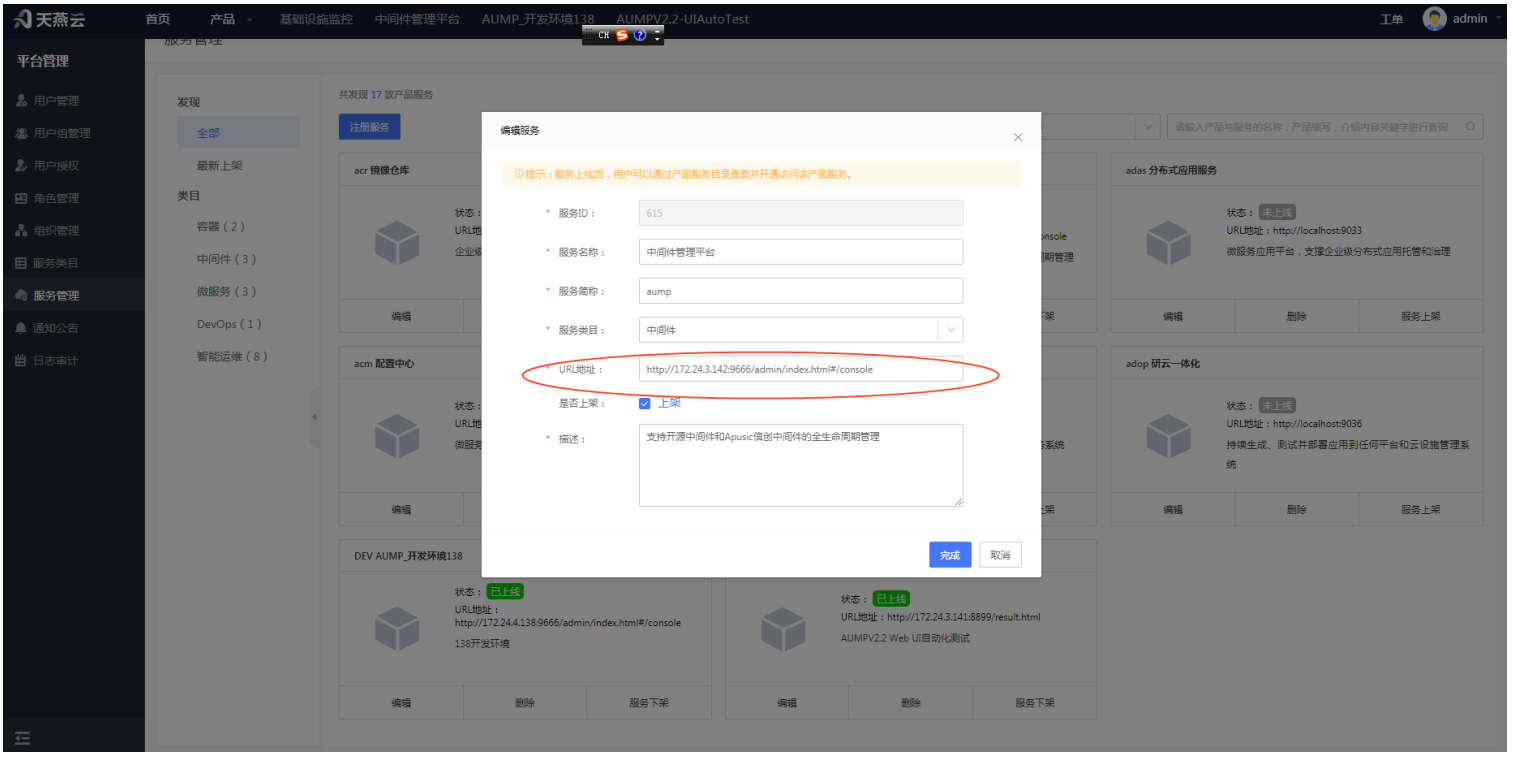
```
cd /opt/aump/aump-admin
chmod 777 bin/*.sh
cd /opt/aump/aump-admin/bin
nohup ./startup.sh > ./aump-admin.log 2>&1 &
```

6.0.5 Web控制台配置服务

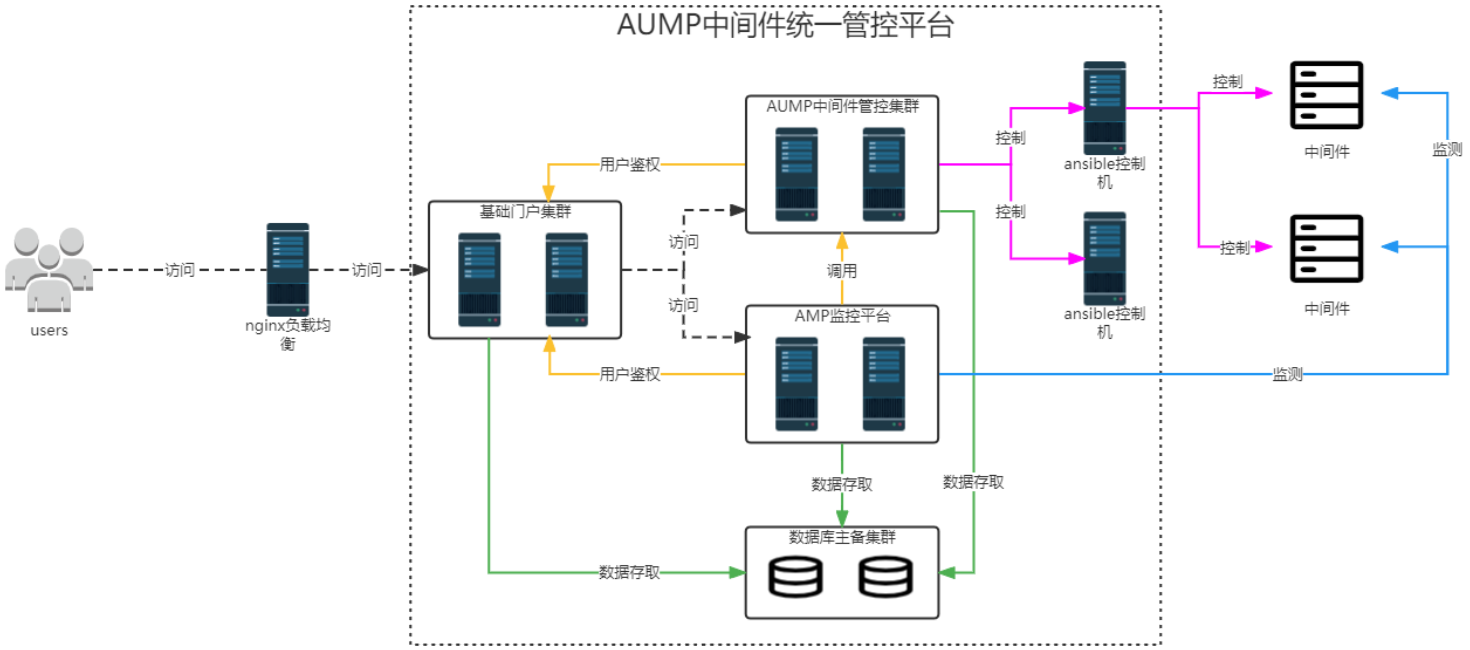
访问IP/aump可访问 控制台。默认用户名/密码为admin/Admin$123456。

进入平台管理的服务管理配置相关URL地址：将localhost改为相应的部署地址，如：http://172.24.3.142:9666/admin/index.html#/console

在AUC的服务管理模块中配置aump 中间件管理平台：



7 高可用部署



对上图的方案描述：

1. 用户的访问通过负载均衡，将所有的访问都分发到三个部件的内部集群中
2. 内部集群之间的访问通过内置的负载均衡机制将访问分散到目标集群
3. 对目标中间件的控制，通过至少两台ansible控制机进行
4. 中间件的检测能力由监控平台实时反馈给用户

此方案的优点：

1. 主要部件都有至少双机备份，并且可以水平扩展更多的机器
2. 控制机的数量没有上限，可以随着中间件数量的增多按比例增加
3. 数据库读写分离，可以扩展成集群，提升可靠性
4. 前置负载均衡可以扩展成多台。如果负载均衡出问题不会影响业务的正常处理。

方案的缺点：

1. 使用session粘滞的负载均衡策略，如果服务处理中断会导致用户需要重新登录一次，并且可能导致表单重新提交
2. 内网如果不适用域名的多IP绑定，就无法使用多个前置负载均衡。在这种情况下如果负载均衡出问题，需要重启才能正常使用。

整个方案除了负载均衡受具体访问方式限定之外，其余部件都没有单点故障存在。

关于负载均衡的分析如下：

1. 在局域网用IP直接访问AUMP服务的话，无法在负载均衡出现故障的情况下自动恢复，但使用虚拟IP方案代价过大。
2. 建议局域网内部也使用域名访问AUMP服务，并为域名绑定多个IP地址，实现负载均衡单点的故障消除。

7.1 Nginx高可用

通过nginx+keepalived实现nginx的高可用，提供外部虚拟ip进行访问，若主nginx宕机失败后，将切换到从nginx，从而实现nginx的高可用。 新建检查nginx时候运行的脚本文件。 在/root目录下创建检查nginx是否运行的脚本，两台服务器均需要进行操作。

```
vi /root/check_nginx.sh
    if [ $(ps -C nginx --no-header |wc -l) -eq 0 ];then
        /usr/local/nginx/sbin/nginx                    #启动nginx
        sleep 2                                        #等待nginx完全启动
    fi
    if [ $(ps -C nginx --no-header |wc -l) -eq 0 ];then
        killall keepalived
    fi
```

修改检查文本为可执行文件

```
chmod +x /root/check_nginx.sh
```

修改keepalived的配置文件 在/etc/keepalived目录下添加keepalived.conf， 若存在，直接进行修改。 添加主节点的keepalived配置

```
    global_defs {
        router_id nginx_01                             #名称
    }
```

vrrp_script check_nginx { script "/root/check_nginx.sh" interval 2 #每2秒检测一次nginx的运行状态 weight -20 #每失败一次，优先级减少20 }

vrrp_instance vrrptest { state MASTER
interface ens160 #选择使用的网卡，保持一致 virtual_router_id 100 #分组标记 mcast_src_ip 172.24.4.110 #本地实际的ip priority 150 #优先级，主节点比从节点的值大一些 advert_int 1 #两台服务器的心跳间隔 authentication { auth_type PASS auth_pass 1111 } track_script { #检查nginx的脚本，和上面的script check_nginx脚本结合使用 check_nginx } virtual_ipaddress { #两台服务器共用的虚拟vip 172.24.4.166 } }

从节点的keepalived的配置基本不变 修改router_id名称， mcast_src_ip本地实际的ip, priority比master小一些，state 修改为BACKUP

```
    global_defs {
        router_id nginx_01                             #名称
    }

    vrrp_script check_nginx {
     script "/root/check_nginx.sh"
     interval 2                                    #每2秒检测一次nginx的运行状态
     weight -20                                         #每失败一次，优先级减少20
    }
```

vrrp_instance vrrptest { state BACKUP
interface ens160 #选择使用的网卡，保持一致 virtual_router_id 100 #分组标记 mcast_src_ip 172.24.4.110 #本地实际的ip priority 100 #优先级，从节点比主节点的值小一些 advert_int 1 #两台服务器的心跳间隔 authentication { auth_type PASS auth_pass 1111 } track_script { #检查nginx的脚本，和上面的script check_nginx脚本结合使用 check_nginx } virtual_ipaddress { #两台服务器共用的虚拟vip 172.24.4.166 } }

关闭防火墙
从主节点启动keepalived

service keepalived start

通过访问nginx的80端口，可以实现访问。
使用命令关掉主服务器上的nginx，发现keepalived的虚拟ip转移到从节点，从节点可以继续访问。 至此nginx+keepalived部署完毕，目前nginx+keepalived部署方案网上的方案也很多也可以进行参考。

7.2 AUC高可用

vi /usr/local/AMP/amp-console/conf/application-prod.yml

```
redis:
    timeout: 3600
    sentinel:
      nodes: 172.24.4.110:26379,172.24.4.111:26379,172.24.4.112:26379
master: mymaster

datasource:
  type: com.zaxxer.hikari.HikariDataSource
  url: jdbc:mysql://localhost:3306/amp_console
?
useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC
  username: root
  password: root
```

控制台组件可以部署多个，但是连接的数据库需要是同一个console数据库。 向nginx的nginx.conf配置文件中加入控制台的配置 下面的配置是部署了两个控制台组件的配置，分别加入每个控制台组件部署的ip和端口号，设置nginx代理的端口为80端口。

```
#控制台
upstream console {
    server console_ip_1:9000;
    server console_ip_2:9000;
 }

 server {
                listen       80;
                server_name  localhost;
         client_max_body_size 1024m;
                location / {
                  proxy_pass http://console;
        client_max_body_size 1024m;
                }
    }
```

至此，控制台组件高可用部署完毕。

7.3 AUMP高可用

修改auc/conf/application-prod.yml配置文件 1.修改mysql数据库连接信息

```
    datasource:
      type: com.zaxxer.hikari.HikariDataSource
      url: jdbc:mysql://localhost:3306/aump_admin
    ?
```

```
useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC
      username: root
      password: root
```

aump组件可以部署多个，但是连接的数据库需要是同一个aump_admin数据库。 向nginx的nginx.conf配置文件中加入控制台的配置 下面的配置是部署了两个控制台组件的配置，分别加入每个控制台组件部署的ip和端口号，设置nginx代理的端口为81端口。 #控制台

```
upstream aump {
    server aump_ip_1:9666;
    server aump_ip_2:9666;
 }

 server {
                listen       81;
                server_name  localhost;
         client_max_body_size 1024m;
                location / {
                  proxy_pass http://aump;
        client_max_body_size 1024m;
                }
    }
```

7.4 Ansible高可用

在多台服务器上安装ansible

安装下列顺序创建sql

id,创建人,创建时间,最后更新人,最后更新日期,数据中心id（数据中心表查询相关id）,安装了ansible的主机（并有相关的ansible脚本）和executor-agent代理执行器,端口号,密码,用户名

执行语句(根据自己的需要高可用的ansible配置)

```
INSERT INTO `excutor_host` VALUES ('1', 'admin', '2022-11-10 14:54:57', 'admin', '2022-11-10
14:55:04', '1129', '172.24.4.138', '8088', 'Apusic1qazXSW@', 'root');

INSERT INTO `excutor_host` VALUES ('2', 'admin', '2022-11-20 14:54:05', 'admin', '2022-11-20
14:54:05', '1129', '172.24.4.63', '8088', 'Apusic1qazXSW@', 'root');
```

7.5 AMP高可用

参考amp高可用部署手册

7.6 Redis/MySQL高可用

参考相关文档高可用部署方案。

7.7 Minio高可用

分布式部署 分 minio 为了提供服务可用性，可以采用分布式高可用方式部署minio,采用纠删码保证数据可用性。 分布式Minio单租户存在最少4个盘最多16个盘的限制（受限于纠删码）。这种限制确保了Minio的简洁，同时仍拥有伸缩性。 分布式Minio里所有的节点需要有同样的access秘钥和secret秘钥，这样这些节点才能建立联接。执行minio server命令之前，先将access秘钥和 secret秘钥export成环境变量。 分布式Minio使用的磁盘里必须是干净的，里面没有数据。 分布式Minio里的节点时间差不能超过3秒，你可以使用NTP 来保证时间一致。 分布式部署示例: 启动分布式Minio实例，8个节点，每节点1块盘，

一共两个节点，每个节点4个挂载硬盘，运行分布式Minio， 在每台服务器节点运行下面的命令.

```
#!/bin/bash
export MINIO_ACCESS_KEY=minioadmin
export MINIO_SECRET_KEY=minioadmin
nohup ./minio server --address ":9010" --console-address ":9001" \
http://192.168.1.1/export1 http://192.168.1.1/export2 \
http://192.168.1.1/export3 http://192.168.1.1/export4 \
http://192.168.1.2/export1 http://192.168.1.2/export2 \
http://192.168.1.2/export3 http://192.168.1.2/export4 > minio.log 2>&1 &
```

export1到export4分别是每个服务器上进行存储的路径驱动目录或硬盘路径。 MINIO_ACCESS_KEY指定的是minio的登录账户 MINIO_SECRET_KEY指定的是minio的登录密码 --address指定的是minio服务端口 --console-address指定的是minio控制台服务端口 不指定上面两个参数时，minio上述两个端口默认使用9000 给minio执行文件，start.sh赋予可执行权限 chmod +x minio start.sh 在分布式部署minio时，每个节点的登录账户MINIO_ACCESS_KEY，登录密码MINIO_SECRET_KEY 分别需要保持一致。 分布式部署问题排查 分

1. 每个节点的账户需要相同，密码需要相同
2. 部署minio的每个服务器节点时间需要进行同步，时差不能超过3秒，可以手动进行同步时间。 真实生产环境可以使用时间同步工具,如NTP等工具 进行同步服务器时间。
3. 每个节点保存数据的目录不能是root文件系统，建议保存在/home目录下。 分布式部署磁盘目录需要特别注意 分

登录linux服务器，执行 df -h命令， 查看服务器磁盘使用情况

```
[root@master01 minio]# df -h
Filesystem               Size  Used Avail Use% Mounted on
devtmpfs                 7.8G     0  7.8G   0% /dev
tmpfs                    7.8G     0  7.8G   0% /dev/shm
tmpfs                    7.8G  1.5M  7.8G   1% /run
tmpfs                    7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/mapper/centos-root  176G   77G   99G  44% /
```

```
/dev/sda1               1014M  226M  789M  23% /boot
/dev/mapper/centos-home   18G  8.6G  8.9G  50% /home
tmpfs                    1.6G     0  1.6G   0% /run/user/0
overlay                  176G   77G   99G  44% /var/lib/docker/overlay2/836937eb78888ad48d1b6b1a9a7cd61b1d17193de53ef0c2ce9f8505fe
overlay                  176G   77G   99G  44% /var/lib/docker/overlay2/eb4be488c339cc95b754e7965c947a9ee27924188f3bce672ec8bbff84
overlay                  176G   77G   99G  44% /var/lib/docker/overlay2/ea9d952ccb00309b826384f3bdd97d908bbce4a51306b9b20386f71bce
shm                       64M     0   64M   0% /var/lib/docker/containers/41df3fdf03c98a39f8a3033bc785370e0364c01a395e09bc08767585
```

对于上面磁盘，不能选择文件系统Filesystem下/dev/mapper/centos-root对应的挂载目录， 即根目录/ , 挂载目录选择该目录会启动失败，不能使用以 root文件系统挂载的磁盘目录。 可以选择文件系统Filesystem下/dev/mapper/centos-home对应的挂载目录，即/home这个目录下保存磁盘数据。 另外尽量选择磁盘容量大的磁盘目录进行保存数据。

Nginx高可用 +keepalived 具体参考后文Nginx高可用

minio高可用

```
upstream minio {
    server minio_ip_1:9010;
    server minio_ip_2:9010;
```

```
    server minio_ip_3:9010;
 }
upstream minioconsole {
    server minioconsole_ip_1:9001;
    server minioconsole_ip_2:9001;
    server minioconsole_ip_3:9001;
 }

 server {
                 listen       9001;
                 server_name  localhost;
         client_max_body_size 1024m;
                 location / {
                   proxy_pass http://minioconsole;
        client_max_body_size 1024m;
                 }
    }

 server {
                 listen       9010;
                 server_name  localhost;
         client_max_body_size 1024m;
                 location / {
                   proxy_pass http://minio;
        client_max_body_size 1024m;
                 }
    }
```

7.8 常见问题和应对

负载均衡出现问题

现象是无法访问，解决方法是重新启动负载均衡。

AUMP服务集群宕机

现象是访问超时，解决方法是重启故障服务，并增加该服务集群的数量。

访问出现500错误

现象是服务内部出现问题，如果是暂时出现则表示服务已经自动恢复，如果一定出现，通常是BUG。

8 附录：环境组件安装

8.1 安装JDK

进入Oracle官网(https://www.oracle.com/technetwork/java/javase/downloads/index.html)，下载对应的JDK版本包进行安装，这里以x86_64架构下的jdk-8u181-linux-x64.tar.gz版本为例介绍安装流程。

1、创建存放java的目录，将jdk安装包解压到特定目录下。

```
mkdir /usr/local/java
tar -zxvf jdk-8u181-linux-x64.tar.gz  -C /usr/local/java
```

2、配置java环境变量。

```
vi /etc/profile
```

3、在/etc/profile里面添加如下内容，修改完成后，wq保存并退出（先按Esc，接着输入 :wq）

```
export JAVA_HOME=/usr/local/java/jdk1.8.0_181
export JAVA_BIN=/usr/local/java/jdk1.8.0_181/bin
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export JAVA_HOME JAVA_BIN PATH CLASSPATH
```

4、配置完成后，输入source profile，再输入java -version命令查看是否配置成功，如果显示java version " jdk1.8.0_181"信息，则表示已经配置成功

```
source /etc/profile
java -version
```

8.2 安装MySQL

AUMP的运行依赖数据库服务，当前支持MySQL，人大金仓等多种类型的关系数据库部署。此处以MySQL为例介绍数据库的安装过程，其他类型数据库请参考数据库厂商产品安装指南进行。

1、首先关闭linux的防火墙，执行命令

```
chkconfig iptables off
```

2、从mysql官网上下载自己适合的mysql版本https://dev.mysql.com/downloads/mysql/5.6.html#downloads，进入mysql官网，进行下载，以下载mysql-5.6.46-linux-glibc2.12-x86_64.tar.gz为例。

3、将下载好的mysql压缩文件放置在linux的/usr/local文件夹下，解压mysql安装包

```
tar zxvf mysql-5.6.46-linux-glibc2.12-x86_64.tar.gz
```

4、将解压后的文件重命名为mysql

```
mv mysql-5.6.46-linux-glibc2.12-x86_64 mysql
```

5、创建mysql用户组及用户

```
groupadd mysql
useradd -r -g mysql mysql
```

6、进入到mysql目录，执行添加MySQL配置的操作

```
cp support-files/my-medium.cnf /etc/my.cnf
或：cp support-files/my-default.cnf /etc/my.cnf
```

是否覆盖？按y 回车 7、编辑/etc/my.cnf文件

```
vi /etc/my.cnf
```

8、在my.cnf文件中添加或者修改相关配置，更改完成后保存退出

```
#These are commonly set, remove the # and set as required.
basedir = /usr/local/mysql
datadir = /usr/local/mysql/data
port = 3306
# server_id = .....
socket = /tmp/mysql.sock
character-set-server = utf8
skip-name-resolve
log-err = /usr/local/mysql/data/error.log
pid-file = /usr/local/mysql/data/mysql.pid
```

9、在mysql当前目录下设定目录的访问权限（注意后面的小点，表示当前目录）

```
chown -R mysql .
chgrp -R mysql .
scripts/mysql_install_db --user=mysql
chown -R root .
chown -R mysql data
```

10、上面第三步执行可能会出现下面的错误

```
[root@localhost mysql-mult]# ./scripts/mysql_install_db  --defaults-file=conf/3306my.cnf
FATAL ERROR: please install the following Perl modules before executing
./scripts/mysql_install_db:
```

11、解决方法： 安装autoconf库

```
yum -y install autoconf
```

12、初始化数据（在mysql/bin或者mysql/scripts下有个 mysql_install_db 可执行文件初始化数据库），进入mysql/bin或者mysql/scripts目录下，执行下面命令

```
./mysql_install_db --verbose --user=root --defaults-file=/etc/my.cnf --
datadir=/usr/local/mysql/data --basedir=/usr/local/mysql
```

13、启动mysql，进入/usr/local/mysql/bin目录，执行下面命令

```
./mysqld_safe --defaults-file=/etc/my.cnf --socket=/tmp/mysql.sock --user=root
```

注意，如果光标停留在屏幕上，表示启动成功，需要我们先关闭shell终端，再开启一个新的shell终端，不要执行退出操作。如果出现 mysql ended这样的语句，表示Mysql没有正常启动，您可以到log中查找问题。 14、设置开机启动，新开启shell中断后，进入mysql目录，执行下面命令

```
cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld
cp /usr/local/mysql/support-files/mysql.server /etc/rc.d/init.d/mysql
chmod 700 /etc/init.d/mysql
chkconfig --add mysqld
chkconfig --level 2345 mysqld on
chown mysql:mysql -R /usr/local/mysql/
```

15、重启操作系统

```
reboot
```

16、查看mysql状态

```
service mysqld status
```

17、添加远程访问权限

（1） 添加mysql命令

ln -s /usr/local/mysql/bin/mysql /usr/bin

（2） 登录mysql，更改访问权限

```
mysql -uroot -p  #密码为空直接回车,运行以下三条命令。
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'yourpassword' with grant option;
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY 'yourpassword' with grant option;
```

（3） 退出mysql

```
exit
```

18、mysql安装完毕

8.3 安装Redis

Web控制台运行需要Redis缓存服务，以下是Redis的简要安装步骤。

1、下载5.05版本在 /usr/local/ 下新建一个 redis 文件夹

```
wget http://download.redis.io/releases/redis-5.0.5.tar.gz
```

2、在 /usr/local/ 下新建一个 redis 文件夹

```
cd /usr/local
mkdir redis
```

3、解压redis-5.0.5.tar.gz安装包

```
tar -zxvf redis-5.0.5.tar.gz
```

4、安装 gcc 环境

```
yum install gcc-c++
```

5、进入解压后的 redis-5.0.5 目录，执行 make 命令

```
cd redis-5.0.5
make
```

6、进入 redis-5.0.5的src 目录后执行 make install命令

```
cd src/
make install
```

7、进入 /usr/local/redis/etc目录，修改 redis.conf 文件

```
cd /usr/local/redis/etc/
vi redis.conf
```

将redis.conf 拷贝到/usr/local/bin下

8、注释掉 bind 127.0.0.1 这一行

```
#bind 127.0.0.1
```

9、将 protected-mode 属性改为 no （关闭保护模式，不然会阻止远程访问；同上，正式服务器项目上线可不修改）

```
protected-mode no
```

10、将 daemonize 属性改为 yes （这样启动时就在后台启动）

```
daemonize yes
```

11、设置密码（可选，建议还是设个密码），修改完成后，保存并退出

```
requirepass redispassword
```

12、在 redis 目录下执行,启动redis，查看redis是否成功启动

```
cd /usr/local/bin/
nohup ./redis-server redis.conf > redis.log 2>&1 &
ps -ef | grep redis
```

8.4 安装Nginx

1.安装ngixn所依赖的插件。

使用下面命令安装gcc, gcc-c++, pcre, pcre-devel, zlib, zlib-devel, openSSL, openssl-devel等nginx依赖的插件。

```
yum install -y gcc gcc-c++ pcre  pcre-devel zlib zlib-devel openssl openssl-devel
```

对上面安装插件的解释说明如下。 安装nginx需要将官网的源码下载进行编译，编译依赖gcc环境，需要安装gcc，gcc-c++。 PCRE(Perl Compatible Regular Expressions) 是一个Perl库，包括 perl 兼容的正则表达式库。nginx 的 http 模块使用 pcre 来解析正则表达式，所以需要在 linux 上安装 pcre 库，pcre-devel 是使用 pcre 开发的一个二次开发库，nginx也需要安装此库。 zlib库（zlib库提供了开发人员的压缩算法，在Nginx的各种模块中需要使用gzip压缩。如同安装PCRE一样，同样需要安装库和它的源代码：zlib和zlib-devel。） OpenSSL库（在Nginx中，如果服务器提供安全网页时则会用到OpenSSL库，我们需要安装库文件和它的开发安装包：openssl和openssl-devel。）

2.安装nginx

先下载好所需的软件安装包，生产环境一般选择稳定版本，这里选择的是nginx-1.16.1.tar.gz, 也可以直接在线安装。

```
cd /usr/local/src           #进入放源码的目录下,将nginx安装包放在该目录下
tar -xvf nginx-1.16.1.tar.gz     #解压
mv nginx-1.16.1  /usr/local/nginx       #将解压包移动到/usr/local/nginx目录
cd /usr/local/nginx
./configure --prefix=/usr/local/nginx    #将nginx所有的资源文件放置在/usr/local/nginx目录下
make
make install
```

若报错在/usr/local/nginx下找不到错误日志logs/err.log，则创建该文件。

```
cd /usr/local/nginx
mkdir logs
touch logs/error.log      #创建错误日志文件
```

启动nginx 确定nginx的80端口没有被其他程序占用，启动nginx程序。

```
/usr/local/nginx/sbin/nginx
```

3.nginx的常用命令

```
./nginx -v                    #查看nginx版本
./nginx                  #启动nginx
ps -ef|grep nginx                #查看nginx是否启动成功
./nginx -s stop          #停止nginx
./nginx -s reload                #重新加载nginx
```

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