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V2.4

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 前置条件

- 每台机器必须支持nc操作（获取实例主机状态需要）

```
 [root@linux-3-142 ~]# nc -v
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: You must specify a host to connect to. QUITTING.
```

安装nc

```
yum install nc
```

- MYSQL需要设置隔离级别
- 安装MYSQL，Redis

2 在主机上安装单机

2.1 概述

通过安装工具进行简单配置可以快速安装aump，局限性为不能够了解每个组件的安装，排查问题需要进入每个组件查看

2.2 资源配置要求

2.2.1 所需的计算资源

最小化的AUMP单机部署，需要下列主机：

- 一个AUMP机器

2.2.2 最低资源要求

单机模式部署，每台机器都需要满足以下最低要求：

表1 - 最低资源要求

机器	操作系统	vCPU	内存	存储	IOPS
AUMP机器	linux主流操作系统	4	16GB	200GB	300

1、当未启用并发多线程 (SMT) 或超线程时，一个 vCPU 相当于一个物理内核。启用后，使用以下公式来计算对应的比例：（每个内核数的线程）× sockets = vCPU。

2、AUMP对磁盘性能非常敏感，建议使用更快的存储速度，特别是 control plane 节点上需要 10 ms p99 fsync 持续时间的 etcd。请注意，在许多云平台上，存储大小和 IOPS 可一起扩展，因此您可能需要过度分配存储卷来获取足够的性能。

2.2.3 网络连接要求

产品组件	产品组件名	端口
aump缓存数据（不落盘）	Redis	6379/26379
统一控制台应用	auc	9000
中间件统一管理服务	aump-admin	9666
运维工单系统	amp-workorder	9014

web监控平台	amp-infra-monitor	9002
amp融合组件	amp-components	9090/9808/9115
告警服务	aalarm-manager	9016
minio软件库	minio	9010/9001
aump控制器	aump-agent	9100

2.3 获取安装程序

2.3.1 如何获取安装程序

- 安装程序文件，联系金蝶天燕营销人员、金蝶天燕的产品服务商在线下载产品介质文件。
- 在安装前，检查产品安装程序文件版本与是实际所购买的是否相符，与本文档适用的产品版本是否一致
- 下载安装工具及安装包 云之家-知识中心 介质下载 知识中心结构说明

目录	说明
pkgs	aump安装包
middleware	已完成aump集成的中间件安装包
sql	aump数据库脚本

2.3.2 安装文件说明

AUMP v2.4完整的产品包括如下安装程序文件，不同CPU架构平台请使用对应的产品安装包。若产品介质名称中不包含x86_64、arm64平台架构的字样，则适合跨平台部署，无需区分。

以下以x86_64的产品安装程序文件为例说明：

表2 - AUMP v2.4产品安装程序文件清单

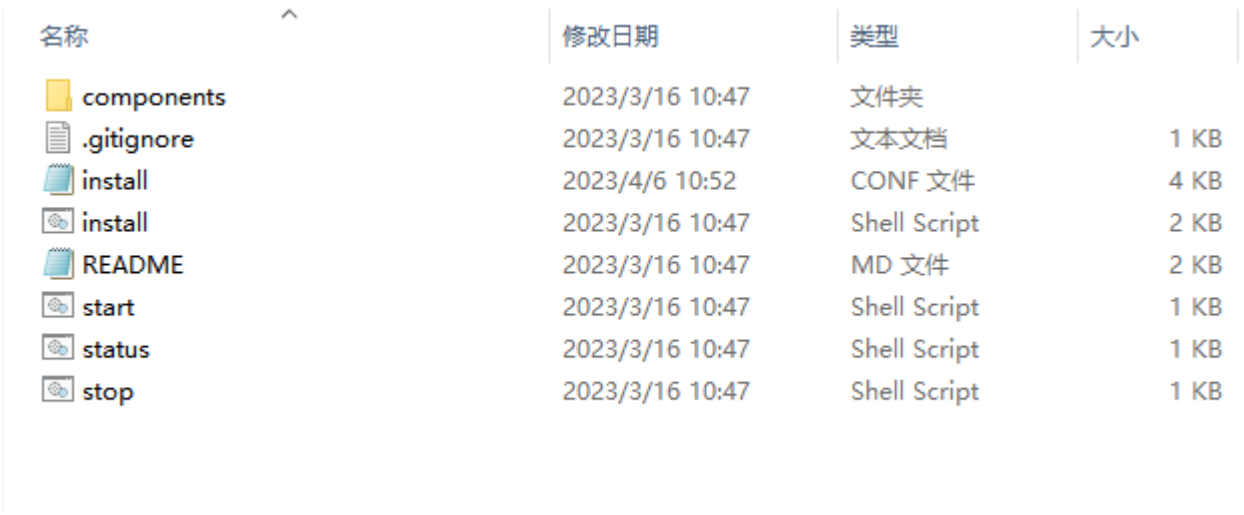
产品组件	文件名	说明
Web控制台	auc-v2.0.tar.gz	统一web控制台应用
中间件管理	aump-admin-2.4.tar.gz	中间件管理后台
Webshell	web-shell-1.0.1.tar.gz	远程终端
Agent代理	aump-agent.tar.gz	代理执行器

对象存储	minio-amd64.zip	中间件软件仓库文件存储
控制器集成的软件包	packagefiles.tar.gz	如AAS v9、ADMQ等
快速安装工具	installer-master.tar.gz	快速安装工具
监控服务	amp-infra-monitor-v3.4.tar.gz	监控服务web应用
监控服务	amp-components.linux-amd64.zip	监控组件
告警服务	aalarm-manager_SE-v1.2.tar.gz	告警服务web应用
管控台	amp-workorder-v3.4.tar.gz	工单系统

2.4 安装AUMP产品

现在通过installer-master安装工具安装aump

1.解压installer-master.tar.gz，解压后如下



2.4.1 安装AUMP

2.4.2 安装准备

初始化了相关数据库脚本

aalarm-create	2023/3/20 10:54	SQL Text File	21 KB
aalarm-initial	2023/3/20 10:54	SQL Text File	2 KB
amp-infra-monitor-create	2023/3/20 10:54	SQL Text File	79 KB
amp-infra-monitor-initial	2023/3/20 10:54	SQL Text File	6,276 KB
amp-workorder-create	2023/3/20 10:54	SQL Text File	10 KB
amp-workorder-initial	2023/3/20 10:54	SQL Text File	4 KB
auc-create	2023/3/20 10:54	SQL Text File	20 KB
auc-initial	2023/3/20 10:54	SQL Text File	171 KB
aump_admin	2023/3/20 10:54	SQL Text File	335 KB

1.将AUMP组件安装包放置在/opt/aump/pkgs目录下

名称	修改日期	类型	大小
installer-master	2023/3/16 10:47	文件夹	
license	2023/3/20 10:55	文件夹	
mysql	2023/3/20 10:54	文件夹	
aalarm-manager_SE-v1.2.tar	2023/3/20 10:52	WinRAR 压缩文件	118,485 KB
amp-components.linux-amd64	2023/3/20 10:52	WinRAR ZIP 压缩...	30,488 KB
amp-infra-monitor-v3.4.tar	2023/3/20 10:53	WinRAR 压缩文件	1,131,545...
amp-workorder-v3.3.tar	2023/3/20 10:53	WinRAR 压缩文件	76,229 KB
auc-v2.0.tar	2023/3/20 10:53	WinRAR 压缩文件	102,799 KB
aump-admin-2.0	2023/3/21 14:27	WinRAR ZIP 压缩...	108,404 KB
minio-amd64	2023/3/20 10:53	WinRAR ZIP 压缩...	30,413 KB
web-shell-1.0.1	2023/3/20 10:53	WinRAR ZIP 压缩...	38,206 KB

2.将AUMP license文件放置在/opt/aump/pkgs/license目录下

2.4.3 修改安装参数

1.修改install.conf中的安装参数

- 修改数据库连接参数：

```
db_type=MYSQL
db_dsn=172.24.5.113:3306
db_user=root
db_password=root
```

- 修改Redis连接参数

```
Redis_type=standalone
Redis_port=6379
Redis_host=172.24.5.113
```

安装包工具配置

```
#【安装包】
package_path=/opt/aump/pkgs

#【节点配置】
# 安装方式，支持standalone
# 安装路劲
#
# 样例:
# install_type=standalone
# install_dir=/opt/amp
#
install_type=standalone
install_dir=/opt/aump/v2.3

#【数据库配置】
# 数据库类型、连接DSN、schema
# 类型支持MYSQL、dm、kingbasees、tdsql
#
# MySQL配置样例:
# db_type=MYSQL
# db_dsn=172.24.5.113:3306
# db_user=root
# db_password=root
#
# 达梦配置样例:
# db_type=dm
# db_dsn=127.0.0.1:5236
# db_schema=public
# db_user=root
# db_password=root
```

```
#
# 金仓配置样例:
# db_type=MYSQL
# db_dsn=127.0.0.1:54321
# db_schema=public
# db_user=root
# db_password=root

db_type=MYSQL
db_dsn=172.24.5.113:3306
db_user=root
db_password=root

#【Redis配置】
# Redis类型，支持standalone
# Redis节点等
#
# 单机样例:
# Redis_type=standalone
# Redis_port=6379
# Redis_host=172.24.5.113
#

Redis_type=standalone
Redis_port=6379
Redis_host=172.24.5.113

# 【auc配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
auc_enable=true
auc_port=9000
auc_database=auc
auc_url=http://localhost:9000

# 【workorder配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
```

```
workorder_enable=true
workorder_port=9014
workorder_database=amp_workorder
workorder_url=http://localhost:9014

# 【aalarm配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
aalarm_enable=true
aalarm_port=9016
aalarm_database=aalarm_manager
aalarm_url=http://localhost:9016
aalarm_license=/opt/aump/pkgs/license/license-alm.xml

# 【amp-infra-monitor配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
monitor_enable=true
monitor_port=9002
monitor_database=amp_monitoring
monitor_url=http://localhost:9002
monitor_license=/opt/aump/pkgs/license/license-amp.xml

# 【amp-components配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
components_enable=true
components_url=http://localhost:9808
# 【aump-admin配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
admin_enable=true
admin_port=9666
admin_database=aump_admin
admin_url=http://localhost:9666
admin_license=/opt/aump/pkgs/license/license-aump.xml

# 【webshell配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
webshell_enable=true
```

```
webshell_port=9999
webshell_url=http://localhost:9999

# 【minio配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
minio_enable=true
minio_console_port=9001
minio_server_port=9010
minio_data_path=/opt/aump/minio/data
minio_url=http://localhost:9010

# 【executor-agent配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
executor_enable=true
executor_port=9100
executor_url=http://localhost:9100
```

2.4.4 安装

```
./install.sh
```

2.4.5 启动

```
./start.sh
```

2.4.6 查看状态

```
./status.sh
```

2.4.7 上传amp采集器到minio

（需要minio启动）

```
./uploadexporter.sh
```

2.4.8 停止

（安装工具支持一键停止所有组件，安装的时候不用执行）

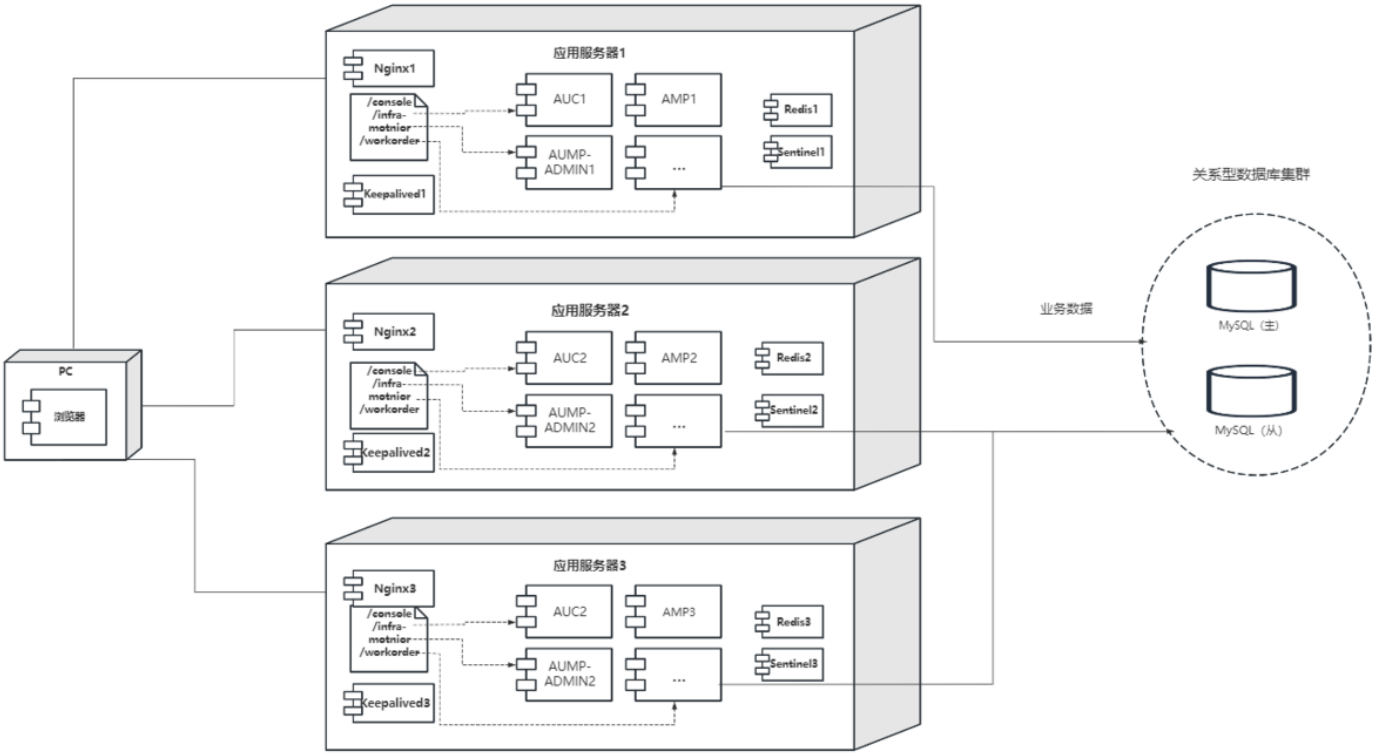
```
./stop.sh
```

3 在主机上安装集群

3.1 概述

安装集群是通过nginx+keepalive对相关组件实现高可用的方式实现，相关的nginx的高可用可以参考附录

3.2 集群部署架构



3.3 资源配置要求

3.3.1 所需的计算资源

部署集群，需要下列主机：

- 3个AUMP机器

3.3.2 最低资源要求

集群模式部署，每台机器都需要满足以下最低要求：

表1 - 最低资源要求

机器	操作系统	vCPU	内存	存储	IOPS
AUMP机器	linux主流操作系统	4	16GB	200GB	300
AUMP机器	linux主流操作系统	4	16GB	200GB	300
AUMP机器	linux主流操作系统	4	16GB	200GB	300

1、当未启用并发多线程 (SMT) 或超线程时，一个 vCPU 相当于一个物理内核。启用后，使用以下公式来计算对应的比例：（每个内核数的线程）× sockets = vCPU。

2、AUMP对磁盘性能非常敏感，建议使用更快的存储速度，特别是 control plane 节点上需要 10 ms p99 fsync 持续时间的 etcd。请注意，在许多云平台上，存储大小和 IOPS 可一起扩展，因此您可能需要过度分配存储卷来获取足够的性能。

3.3.3 网络连接要求

服务器	产品组件	产品组件名	端口
192.168.1.1	aump缓存数据（不落盘）	Redis	6379/26739
	统一控制台应用	auc	9000
	aump高可用，对外暴露端口	nginx+keepalive	80
	中间件统一管理服务	aump-admin	9666
	运维工单系统	amp-workorder	9014
	web监控平台	amp-infra-monitor	9002
	站点监控	amp-blackbox	9115
	告警服务	aalarm-manager	9016
	amp融合组件	amp-components	9090/9808/9115
	minio软件库	minio	9010/9001
	aump控制器	aump-agent	9100
192.168.1.2	aump缓存数据（不落盘）	Redis	6379/26739
	aump高可用，对外暴露端口	nginx+keepalive	80
	统一控制台应用	auc	9000
	中间件统一管理服务	aump-admin	9666

	运维工单系统	amp-workorder	9014
	web监控平台	amp-infra-monitor	9002
	站点监控	amp-blackbox	9115
	告警服务	aalarm-manager	9016
	amp融合组件	amp-components	9090/9808/9115
	minio软件库	minio	9010/9001
	aump控制器	aump-agent	9100
192.168.1.3	aump缓存数据（不落盘）	Redis	6379/26739
	aump高可用，对外暴露端口	nginx+keepalive	80
	统一控制台应用	auc	9000
	中间件统一管理服务	aump-admin	9666
	运维工单系统	amp-workorder	9014
	web监控平台	amp-infra-monitor	9002
	站点监控	amp-blackbox	9115
	告警服务	aalarm-manager	9016
	amp融合组件	amp-components	9090/9808/9115
	minio软件库	minio	9010/9001
	aump控制器	aump-agent	9100

3.4 安装AUMP产品

安装aump集群也是通过install-master安装工具安装，安装步骤如下

1.安装准备

安装准备

2.安装nginx+keepalived高可用

nginx高可用

3.redis高可用

redis高可用

4.免密配置

```
ssh-keygen -t rsa -P "" -f /root/.ssh/id_rsa
# 样例 ，把免密id拷贝到其他需要安装的服务器，包括本地服务器
ssh-copy-id root@192.168.1.1
ssh-copy-id root@192.168.1.2
ssh-copy-id root@192.168.1.3
```

5.修改install.conf中的安装参数

```
#【安装包】
package_path=/opt/aump/pkgs

#【节点配置】
# 安装方式，支持standalone、cluster
# 安装路劲
#
# 样例:
# install_type=standalone
# install_dir=/opt/amp
#
# install_type=cluster
# install_dir=/opt/amp
# install_nodes=127.0.0.1,127.0.0.2,127.0.0.3
# work_node=127.0.0.1

install_type=standalone
install_dir=/opt/aump/v2.4

#【数据库配置】
# 数据库类型、连接DSN、schema
# 类型支持mysql、dm、kingbasees、tdsql
#
# MySQL配置样例:
# db_type=mysql
```

```
# db_dsn=172.24.5.113:3306
# db_user=root
# db_password=root
#
# 达梦配置样例:
# db_type=dm
# db_dsn=127.0.0.1:5236
# db_schema=public
# db_user=root
# db_password=root
#
# 金仓配置样例:
# db_type=kingbasees
# db_dsn=127.0.0.1:54321
# db_schema=public
# db_user=root
# db_password=root

db_type=mysql
db_dsn=172.24.5.113:3306
db_user=root
db_password=root

#【redis配置】
# redis类型，支持standalone
# redis节点等
#
# 单机样例:
# redis_type=standalone
# redis_port=6379
# redis_host=172.24.5.113
# redis_password=123456
# 哨兵样例:
#redis_type=sentinel
#redis_master=mymaster
#redis_nodes=127.0.0.1:26379,127.0.0.2:26379,127.0.0.3:26379
```

```
#redis_password=123456
redis_type=standalone
redis_port=6379
redis_host=172.24.5.113

# 【auc配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
auc_enable=true
auc_port=9000
auc_database=auc
auc_url=http://localhost:9000

# 【workorder配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
workorder_enable=true
workorder_port=9014
workorder_database=amp_workorder
workorder_url=http://localhost:9014

# 【aalarm配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
aalarm_enable=true
aalarm_port=9016
aalarm_database=aalarm_manager
aalarm_url=http://localhost:9016
aalarm_license=/opt/aump/pkgs/license/license-alm.xml

# 【amp-infra-monitor配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
monitor_enable=true
monitor_port=9002
monitor_database=amp_monitoring
monitor_url=http://localhost:9002
monitor_license=/opt/aump/pkgs/license/license-amp.xml

# 【amp-components配置】
```

```
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
components_enable=true
components_url=http://localhost:9808

# 【aump-admin配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
admin_enable=true
admin_port=9666
admin_database=aump_admin
admin_url=http://localhost:9666
admin_license=/opt/aump/pkgs/license/license-aump.xml

# 【webshell配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
webshell_enable=true
webshell_port=9999
webshell_url=http://localhost:9999

# 【minio配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
minio_enable=true
minio_console_port=9001
minio_server_port=9010
minio_data_path=/opt/aump/minio
minio_url=http://localhost:9010
minio_software_path=/opt/aump/files

# 【aump-agent配置】
# 是否部署、端口、数据库、代理地址：如nginx反向代地址,集群模式必填
aump_agent_enable=true
aump_agent_port=9100
```

3.4.1 安装

```
./install.sh
```

3.4.2 启动

```
./start.sh
```

3.4.3 查看状态

```
./status.sh
```

3.4.4 上传amp采集器到minio

（需要minio启动）

```
./uploadexporter.sh
```

3.4.5 停止

（安装工具支持一键停止所有组件，安装的时候不用执行）

```
./stop.sh
```

4 后续步骤

单机安装和高可用安装都要做如下步骤

4.0.1 多数据中心配置

选择aump_admin数据库,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, '0.0.0.0-0.0.0.0', 'xxx', '内置SSH数据中心', null, null, '2022-12-13 13:52:24', null, '2023-04-13 17:02:15', null, '0', 'SSH');
INSERT INTO `data_center` VALUES ('2', null, '0.0.0.0-0.0.0.0', 'xxx', '内置HTTP数据中心', null, null, '2023-03-13 14:18:38', null, '2023-03-13 14:18:43', null, '0', 'HTTP');
```

SSH模式

在选定机器安装aump-agent（见下文aump-agent代理执行器安装），添加主机时候需要输入目标机器的账号密码,在安装了agent执行器主机 写相关的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aump-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', '9100', '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', '9100', 'Apusic1qazXSW@', 'root');
```

执行excutor_host表，填入agent的相关的信息，

HTTP模式

在目标主机安装aump-agent（见下文aump-agent代理执行器安装），添加主机时候可以不需要输入目标机器的账号密码

4.0.2 aump-agent代理执行器安装

```
tar -zxvf aump-agent-amd64.tar.gz -d /opt/aump
```

启动代理执行器服务程序

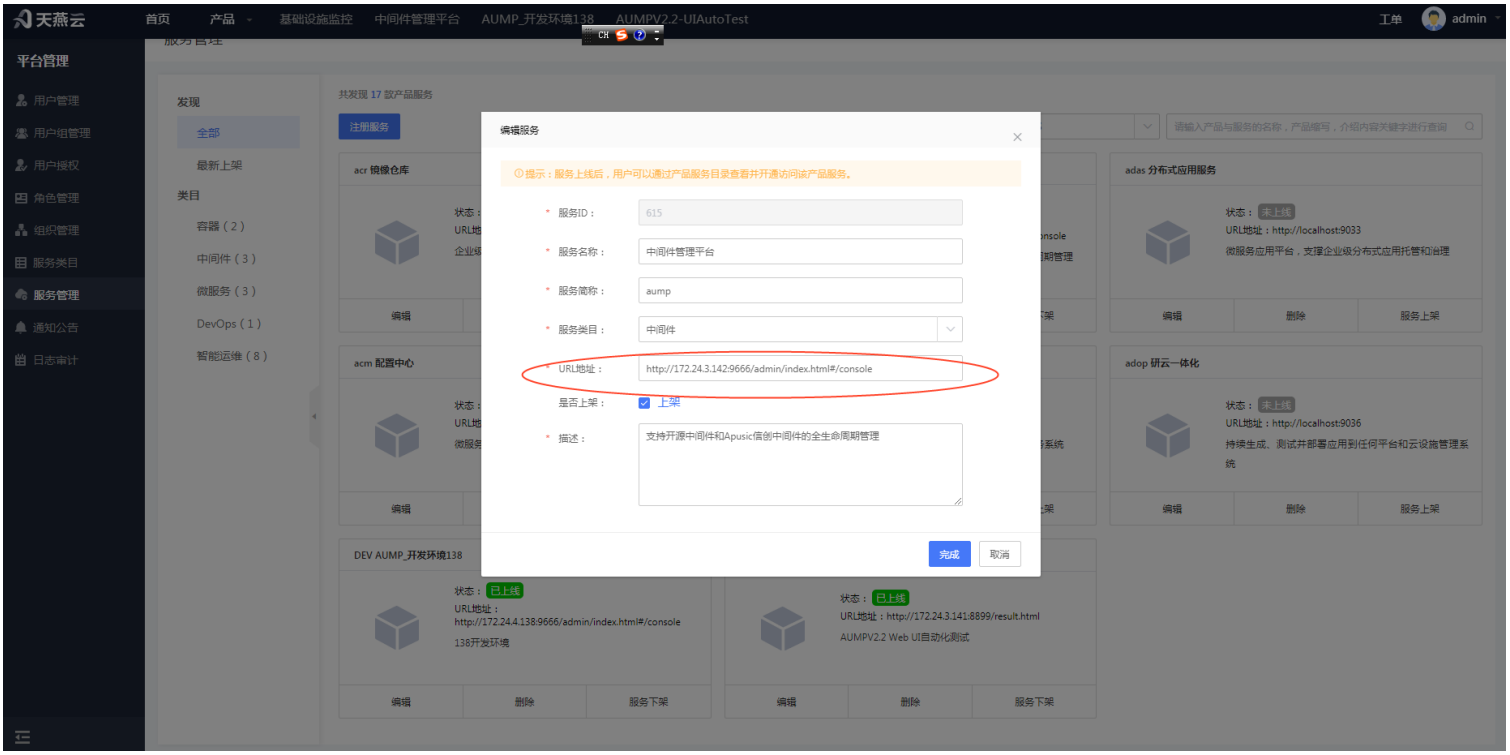
进入安装目录

```
nohup ./aump-agent > aump-agent.logs 2>&1 &
```

启动执行器代理程序，默认使用9100端口

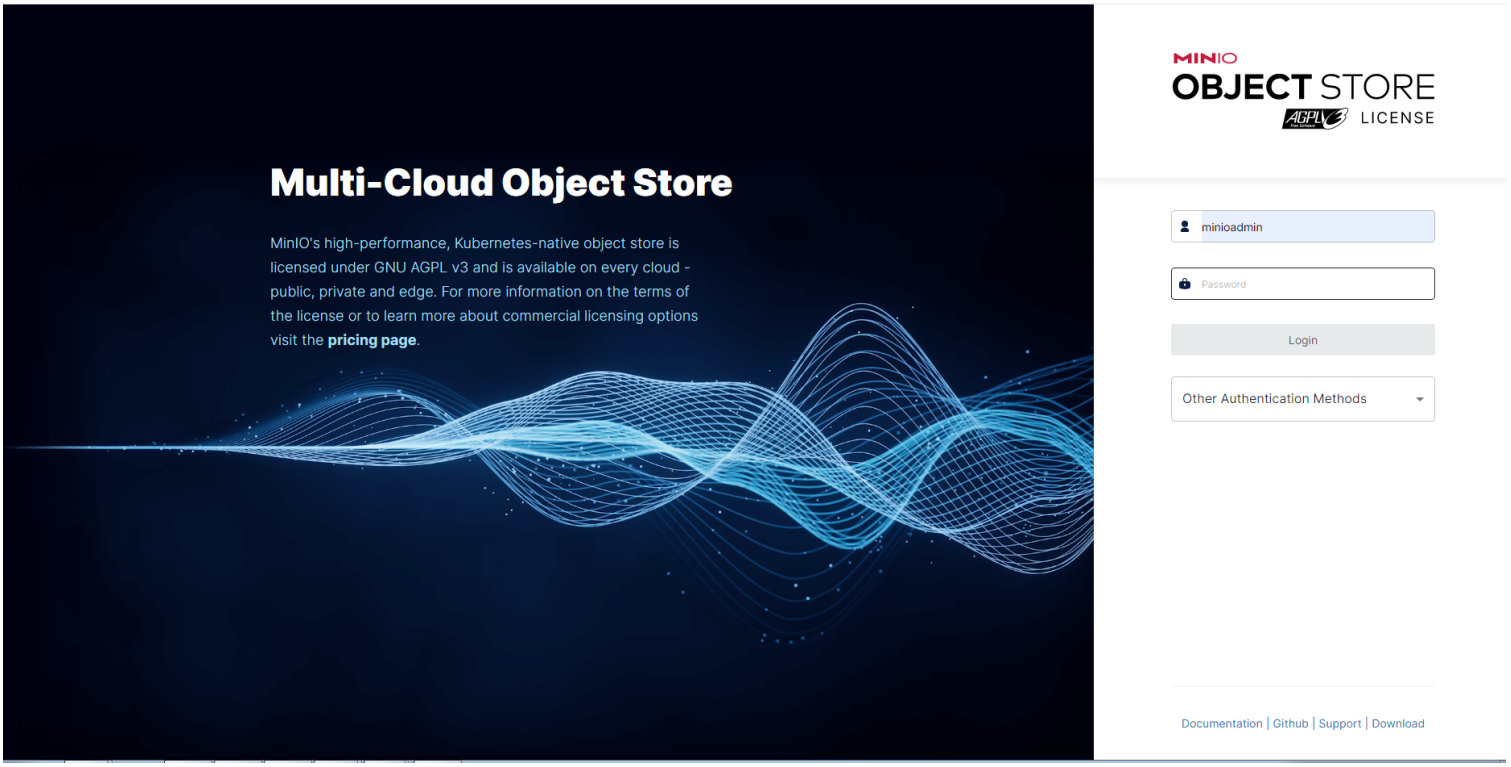
4.0.3 Web控制台配置服务

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,Auc配置aump 中间件管理平台.

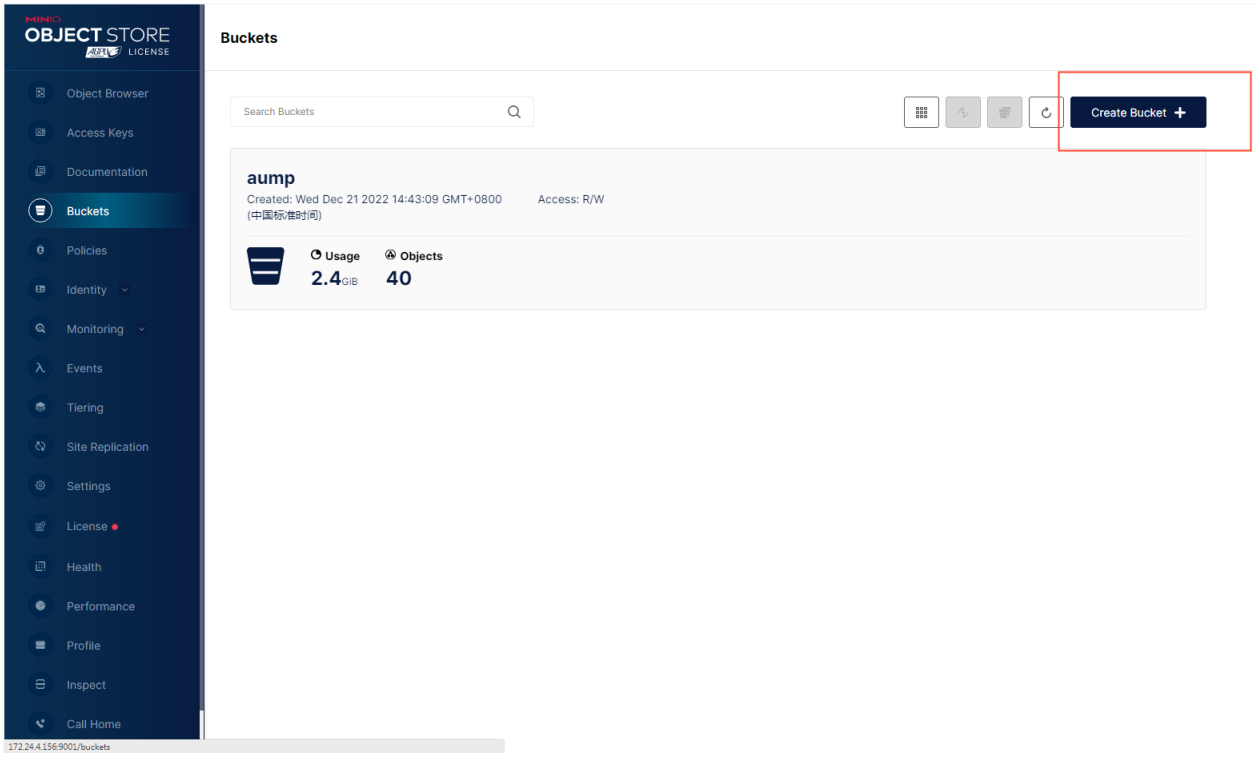


4.0.4 minio配置

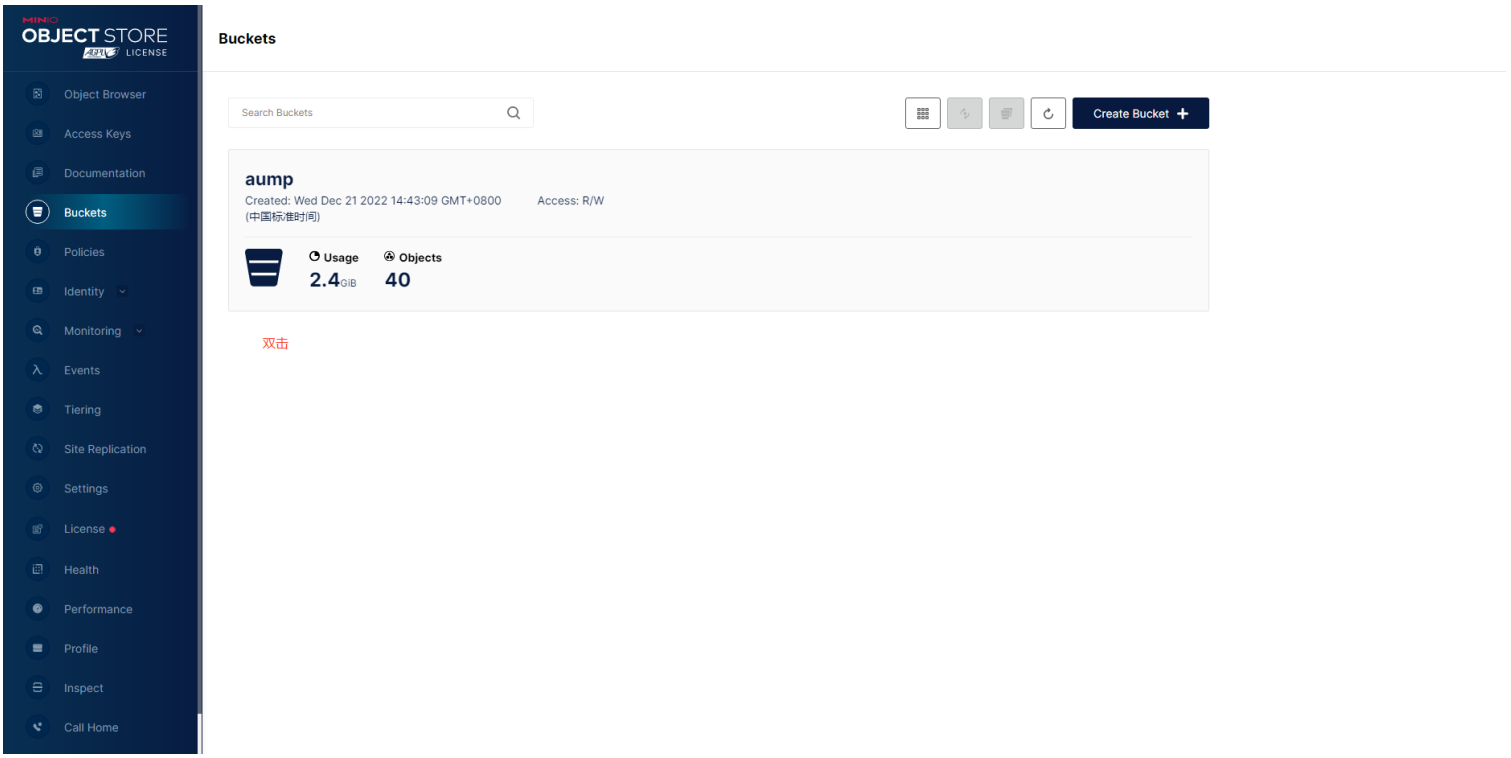
输入minio地址,http://172.24.3.145:9001/login(配置的地址)账号密码为minioadmin

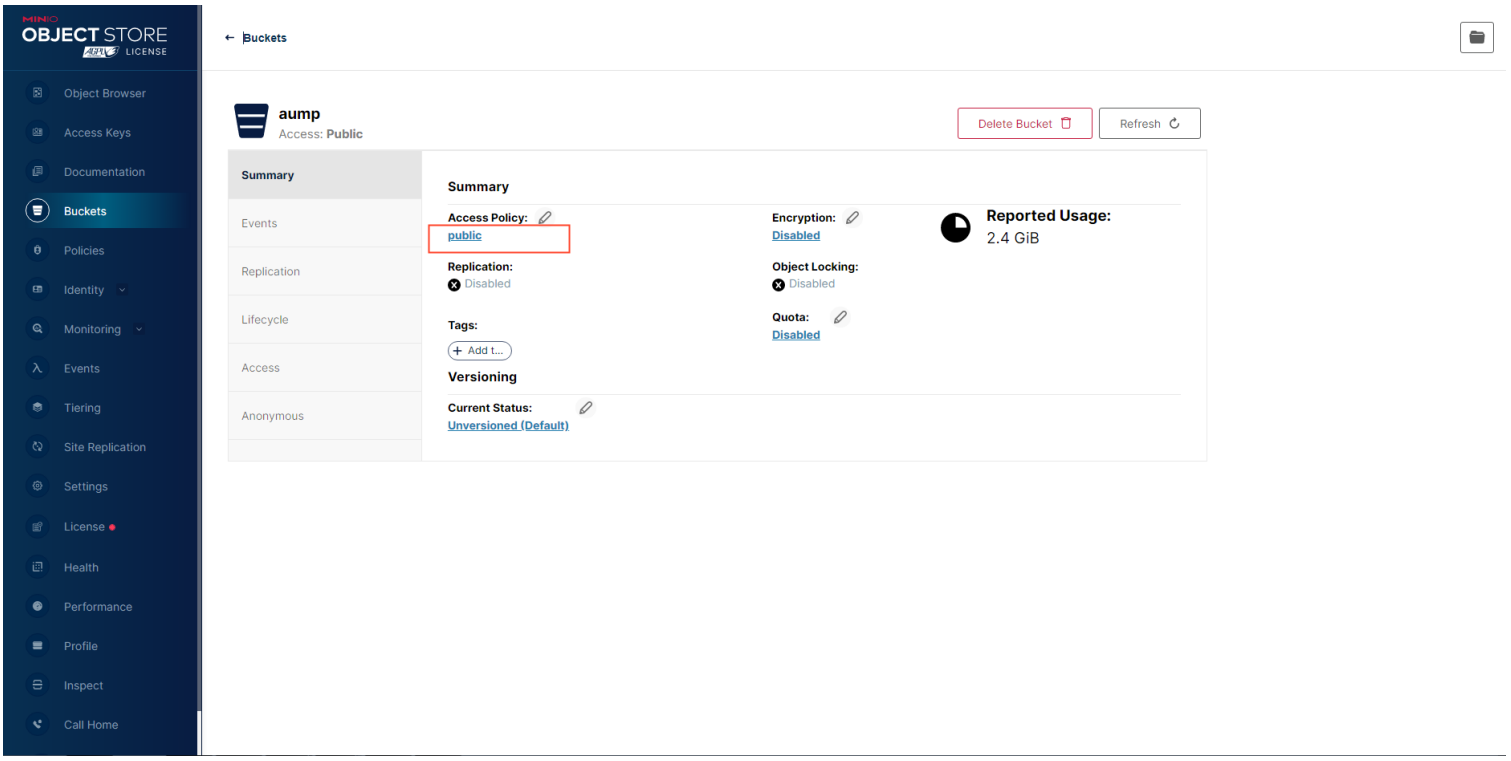


创建相关的bucket名字为aump

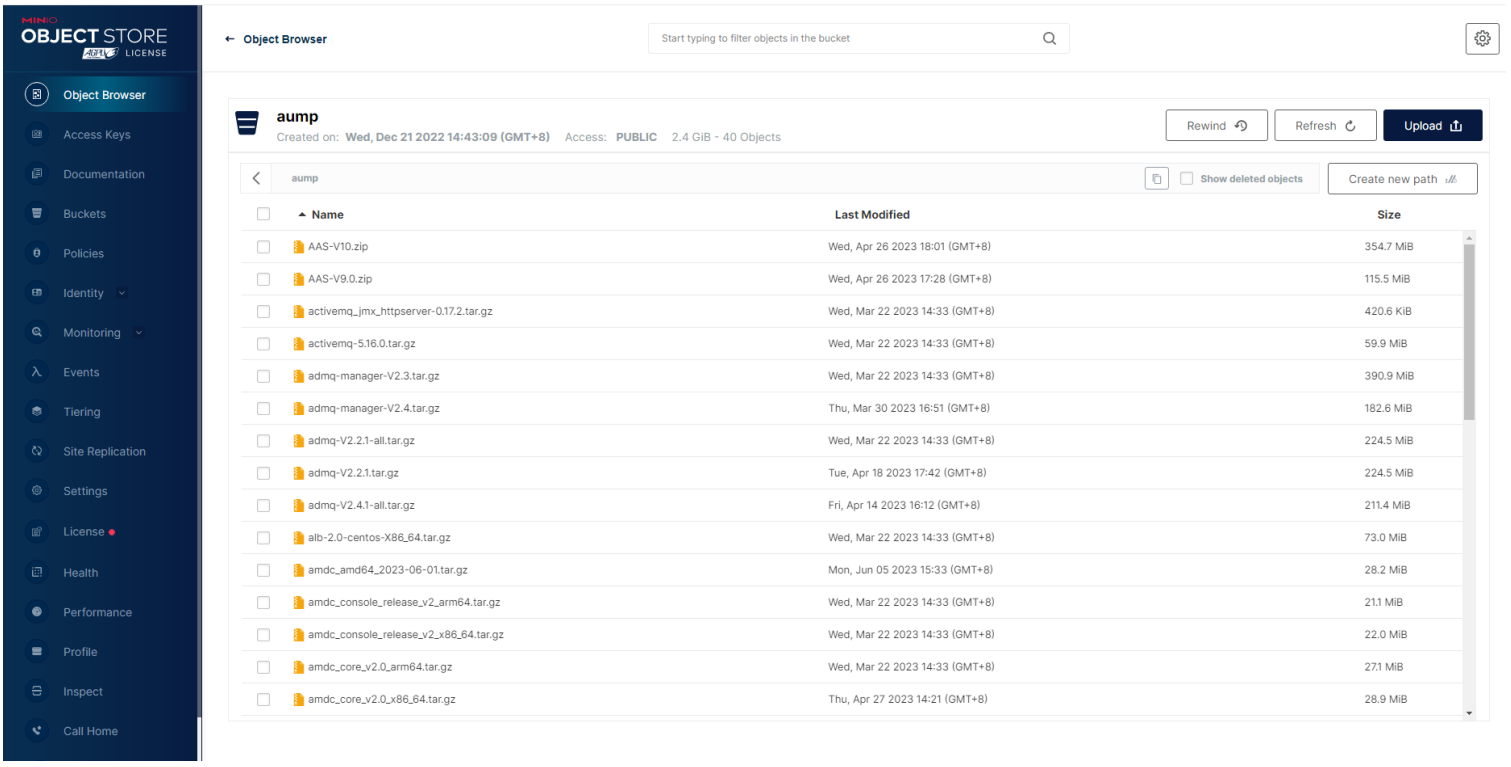


双击bucket将access policy 改为public

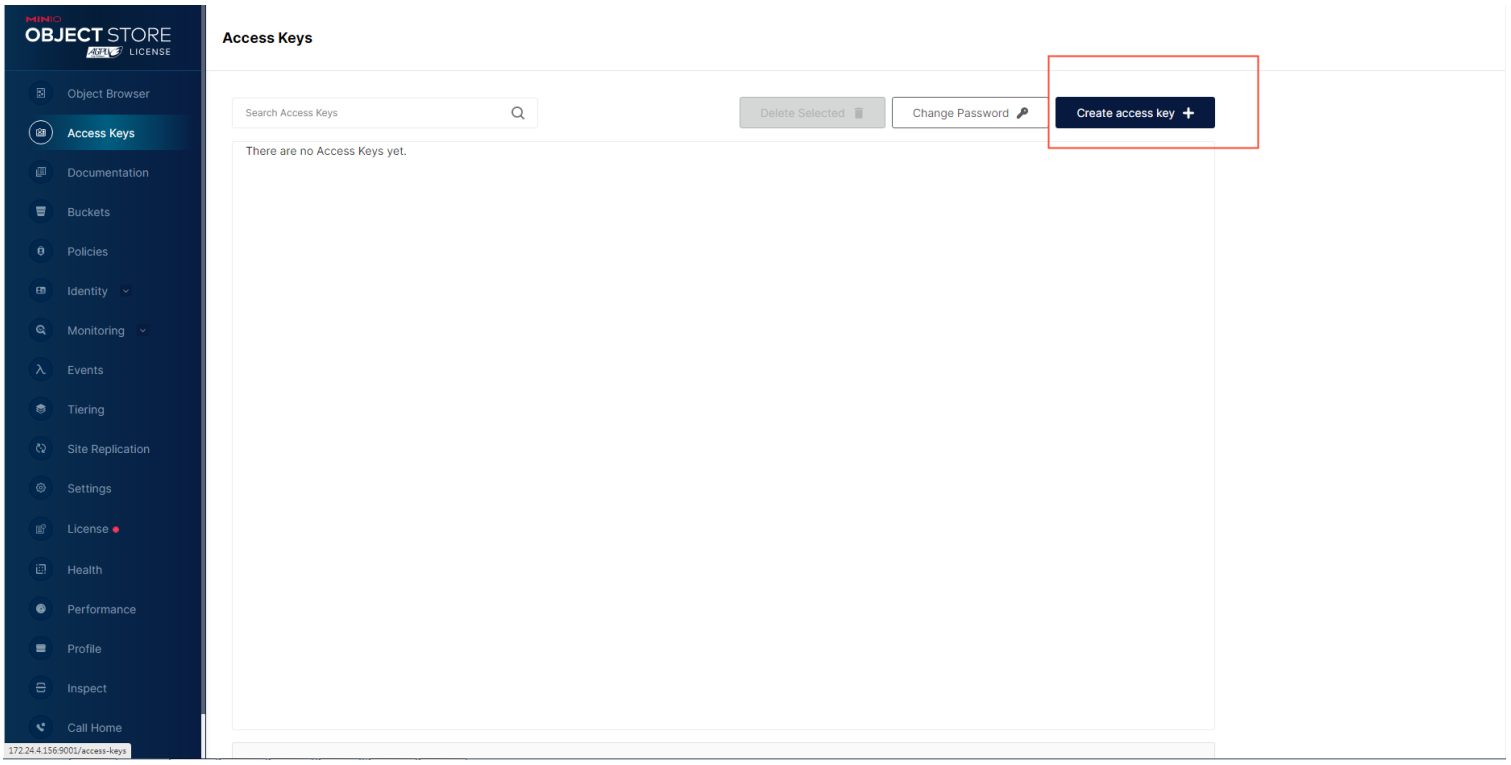




点browse将安装介质的软件包files下的包上传上去



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



5 附录

5.1 在主机上安装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MYSQL下载，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

5.1.1 MYSQL读已提交隔离级别级别设置

登陆到MYSQL

```
mysql -uroot -p123456
```

执行以下命令查看事务隔离参数

```
show variables like 'transaction_isolation';
```

返回:

```
MYSQL> show variables like 'transaction_isolation';

+-----------------------+-----------------+

| Variable_name         | Value           |

+-----------------------+-----------------+

| transaction_isolation | REPEATABLE-READ |

+-----------------------+-----------------+

1row in set (0.00sec)
```

需要手动修改为以下 临时修改:

```
set global transaction isolation level read committed;
set session transaction isolation level read committed;
```

永久修改:

vim /etc/my.cnf --参数位置可能不同

在[MYSQLd]下添加:

```
transaction-isolation = READ-COMMITTED
```

5.1.2 MYSQL高可用最大单数据库连接数

vim /etc/my.cnf --参数位置可能不同
在[MYSQLd]下添加:

max_connections=1000

5.2 在主机上安装JDK

进入Oracle官网，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
```

5.3 在主机安装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
```

5.4 Nginx高可用

安装Nginx

Nginx的版本为1.16.1。Keepalived安装过程如下

```
yum install -y keepalived
```

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

        if [ $(ps -C nginx --no-header |wc -l) -eq 0 ];then
            killall keepalived
        fi
```

```
    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（interface ens192 为网卡信息 ip a 查看）

```
[root@linux-3-62 aump]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state
UP group default qlen 1000
    link/ether 00:50:56:99:9d:82 brd ff:ff:ff:ff:ff:ff
    inet 172.24.3.62/24 brd 172.24.3.255 scope global noprefixroute
ens192
       valid_lft forever preferred_lft forever
    inet 172.24.3.66/32 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe99:9d82/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

填写网卡为 ens192

```
        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 ens192           #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 ens192           #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
                        }
                    }
```

aump相关高可用nginx.conf配置，修改相关ip

```
    #user  nobody;
    worker_processes  1;

    #error_log  logs/error.log;
    #error_log  logs/error.log  notice;
    #error_log  logs/error.log  info;

    #pid        logs/nginx.pid;


    events {
        worker_connections  1024;
    }


    http {
        include       mime.types;
        default_type  application/octet-stream;
        #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
        #                  '$status $body_bytes_sent "$http_referer" '
        #                  '"$http_user_agent" "$http_x_forwarded_for"';
        log_format main '{ "@timestamp": "$time_local", '
                            '"@fields": { '
                            '"uri":"$request_uri",'
                            '"url":"$uri",'
                            '"upstream_addr":"$upstream_addr",'
```

```
                              '"remote_addr": "$remote_addr", '
                              '"remote_user": "$remote_user", '
                              '"body_bytes_sent": "$body_bytes_sent", '
                              '"host":"172.24.4.195",'
                              '"server_addr":"$server_addr",'
                              '"request_time": "$request_time", '
                              '"request_time":"$request_time",'
                              '"status":"$status",'
                              '"request": "$request", '
                              '"request_method": "$request_method", '
                              '"size":$body_bytes_sent,'

'"upstream_time":"$upstream_response_time"'
                              '"http_referrer": "$http_referer", '
                              '"body_bytes_sent":"$body_bytes_sent", '
                              '"http_x_forwarded_for":
"$http_x_forwarded_for", '
                              '"http_user_agent": "$http_user_agent" }
}';

        #access_log  logs/access.log  main;

        sendfile        on;
        #tcp_nopush     on;

        #keepalive_timeout  0;
        keepalive_timeout  65;

        #gzip  on;
        gzip  on;
        gzip_min_length   1k;
        gzip_buffers      4 16k;
        gzip_http_version 1.0;
        gzip_comp_level   3;
        gzip_types        text/plain text/css application/x-javascript
text/xml application/xml application/xml+rss text/javascript
```

```
application/json application/javascript;
        gzip_vary         on;
        proxy_http_version 1.1;
        proxy_set_header Connection "";

        upstream auc {
          #实际安装组件ip地址
          server 172.24.3.62:9000;
          server 172.24.3.63:9000;
          server 172.24.3.64:9000;
        }

        upstream amp {
          #实际安装组件ip地址
          server 172.24.3.62:9002;
          server 172.24.3.63:9002;
          server 172.24.3.64:9002;
        }

        upstream webshell {
          #实际安装组件ip地址
          server 172.24.3.62:9999;
          server 172.24.3.63:9999;
          server 172.24.3.64:9999;
        }

        upstream workorder {
          #实际安装组件ip地址
          server 172.24.3.62:9014;
          server 172.24.3.63:9014;
          server 172.24.3.64:9014;
        }

        upstream component {
          #实际安装组件ip地址
          server 172.24.3.62:9808;
```

```
    server 172.24.3.63:9808;
    server 172.24.3.64:9808;
}

upstream prom {
    #实际安装组件ip地址
    server 172.24.3.62:9091;
    server 172.24.3.63:9091;
    server 172.24.3.64:9091;
}

upstream aalarm {
    #实际安装组件ip地址
    server 172.24.3.62:9016;
    server 172.24.3.63:9016;
    server 172.24.3.64:9016;
}

upstream aumpadmin {
    #实际安装组件ip地址
    server 172.24.3.62:9666 weight=7;
    server 172.24.3.63:9666 weight=1;
    server 172.24.3.64:9666 weight=1;
}
    upstream minio {
        #实际安装组件ip地址
        server 172.24.3.62:9010;
        server 172.24.3.63:9010;
        server 172.24.3.64:9010;
    }
   upstream minioconsole {
         #实际安装组件ip地址
        server 172.24.3.62:9001;
        server 172.24.3.63:9001;
        server 172.24.3.64:9001;
    }
```

```
server {
            listen       80;
            server_name  localhost;

            #charset koi8-r;

            #access_log  logs/host.access.log  main;

            location ^~/ {
                client_max_body_size 2000M;
                proxy_pass http://auc/;
            }

            location ^~/amp/ {
                proxy_pass http://amp/;
            }

           location ^~ /webshell {
              proxy_pass http://webshell;
                  proxy_set_header Host $host;
                  proxy_http_version 1.1;
                  proxy_set_header Upgrade $http_upgrade;
                  proxy_set_header Connection "upgrade";
           }

            location ^~/workorder/ {
                proxy_pass http://workorder/;
            }



            location ^~/prom/ {
                proxy_pass http://prom/;
            }
```

```
                location ^~/aalarm/ {
                    proxy_pass http://aalarm/;
                }


                location ^~/admin {
                    client_max_body_size 2000M;
                    proxy_pass http://aumpadmin;
                }


                error_page   500 502 503 504  /50x.html;
                location = /50x.html {
                    root   html;
                }
        }
        #minio传输地址
        server {
                listen       81;
                server_name  localhost;
                client_max_body_size 2000M;


                location ^~ / {
                   proxy_set_header X-Real-IP $remote_addr;
                   proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
                   proxy_set_header X-Forwarded-Proto $scheme;
                   proxy_set_header Host $http_host;

                   proxy_connect_timeout 300;
                    # Default is HTTP/1, keepalive is only enabled
in HTTP/1.1
                   proxy_http_version 1.1;
                   proxy_set_header Connection "";
```

```
            chunked_transfer_encoding off;

            proxy_pass http://minio;
        }

    }
#minio控制台端口
server {
    listen       82;
    server_name  localhost;
    client_max_body_size 2000M;

    location ^~/ {
        proxy_pass http://minioconsole;
    }
    location /ws/objectManager {
                proxy_http_version 1.1;
                proxy_set_header Origin '';
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_pass http://minioconsole;
            }
}

server {
    listen       83;
    server_name  localhost;
    client_max_body_size 2000M;

    location ^~/ {
        proxy_pass http://component/;
    }
}


# HTTPS server
```

```
        #
        #server {
        #    listen       443 ssl;
        #    server_name  localhost;

        #    ssl_certificate      cert.pem;
        #    ssl_certificate_key  cert.key;

        #    ssl_session_cache    shared:SSL:1m;
        #    ssl_session_timeout  5m;

        #    ssl_ciphers  HIGH:!aNULL:!MD5;
        #    ssl_prefer_server_ciphers  on;

        #    location / {
        #        root   html;
        #        index  index.html index.htm;
        #    }
        #}

    }
```

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。

注：替换ngin配置文件，需要先关闭keepalived，在重新启动nginx

5.5 Redis高可用

参考相关文档高可用

redis高可用

5.6 mysql高可用

mysql高可用自定义

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