Apusic 金蝶天燕

生成时间：2026-05-19

安装手册

金蝶Apusic监控平台V3.4

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 简介

金蝶Apusic监控平台软件（Apusic Monitor Platform ,以下简称AMP）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一体化云原生监控平台产品。 AMP从业务系统视角出发，对服务器、网络设备、存储、数据库、中间件、基础云服务、业务应用系统进行一体化、自动化、智能化的全面的监控和运维。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。 AMP使IT基础架构真正成为保障业务服务水平的、可管理、可控制的业务平台，构建业务人员和IT运维及管理人员的联系纽带，帮助组织快速应对外部变化给IT基础设施和业务应用带来的冲击和挑战。

2 范围和读者

本手册介绍AMPV3.4产品安装过程说明，主要适用于产品技术顾问、产品使用用户等。

3 文档导航

章节 内容概述

1. 前言 文档范围，约定内容
2. 安装环境要求 AMP产品安装所需的软硬件环境
3. 安装概述 AMP产品安装部署说明
4. 安装Web控制台 Web控制台安装过程说明
5. 安装基础设施监控服务 基础设施监控服务安装过程说明
6. 安装运维工单系统 运维工单系统安装过程说明
7. 安装容器云监控服务 容器云监控服务安装过程说明
8. 集群高可用安装 AMP产品高可用安装说明
9. 自动化安装 AMP产品Ansible自动化安装说明
10. 附录 运行AMP依赖的其他环境组件的安装说明

4 1.4 约定与术语

一些约定的缩略词诠释：

• AMP

金蝶Apusic监控平台（Apusic Monitor Platform）

• Kubernetes

CNCF社区开源的容器编排管理平台

• M3DB

Uber开源开源的分布式时序数据库

• Prometheus

CNCF社区开源的云原生监控项目

• Ansible

基于Python的自动化运维工具

5 第2章 安装环境要求

6 2.1 配置要求

安装AMPV3.4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	16G或以上
硬盘	可用空间1T或以上
浏览器	FireFox 21及以上、Chrome 23及以上、IE 10及以上

表 2 1 软件及操作系统环境要求

7 2.2 推荐配置

安装AMP产品的推荐的配置见下表：

资源环境	要求
操作系统	Linux
CPU	Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz或以上
MySQL	5.6或以上
Redis	5.0或以上
内存	16G或以上
硬盘	可用空间1T或以上
浏览器	Chrome 60及以上、FireFox 21及以上

表2-2 软件及操作系统环境推荐配置

8 第3章 安装概述

9 3.1 产品介质说明

当您购买产品后，从我们邮寄给您的安装光盘中拷贝所有产品介质，或者您可以联系您的产品供应商获取产品介质。

AMP V3.4产品包括如下10个文件，请在安装前逐一检查。不同平台请使用对应用的产品安装包，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。

组件名称	文件名	说明
Web控制台	amp-console-v3.4.tar.gz	统一web控制台应用
Web监控平台	amp-infra-monitor-v3.4.tar.gz	基础设施监控服务的web监控平台组件
告警服务器	aalarm-manager_SE-v1.2.tar.gz	基础设施监控服务的告警服务器组件，Prometheus监控引擎的报警服务器组件，用于监控报警信息的通知
运维工单系统	amp-workorder-v3.4.tar.gz	运维工单系统，运维工单的流程闭环管理
容器云监控	amp-cloud-monitor-v3.4.tar.gz	容器云监控服务，用于监控k8s集群
插件	amp-monitor-kubernetes.linux-amd64.tar.gz	容器云监控服务的k8s插件
推送网关服务器	amp-pushgateway-amd64.tar.gz	基础设施监控服务的服务器组件，用于推送监控数据
组件	amp-components-v3.4.tar.gz	基础设施监控服务器组件(包含prom_agent，nettopo，prometheus，blackbox四个组件功能)

表3-1 AMP V3.3产品组件说明

10 3.2 产品组件关系

AMP V3.4产品中各组件调用关系如下图3-1所示

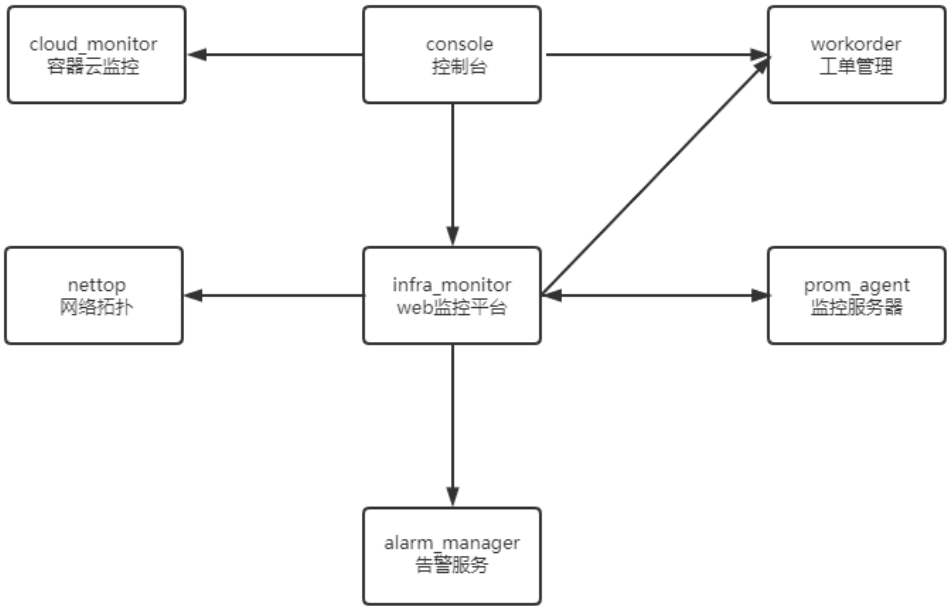


图 3-1 AMP V3.4各组件关系调用图 AMP V3.4各组件的作用如下所示。

- Web控制台：用户，权限，角色，项目，服务，日志审计管理等功能。
- Web监控平台：实现具体监控功能的管理，如创建监控任务，报警策略等。
- 监控服务器：包括监控agent服务，prometheus监控引擎，实现具体的监控功能。
- 告警服务器：对告警信息进行分组处理，产生告警通知。
- 网络拓扑：用于展示网络中各网络设备之间的关系。
- 运维工单系统：提交表单，跟进问题，反馈处理结果。
- 容器云监控：用于监控k8s集群。

11 3.3 产品安装说明

本文档以在Linux x86_64环境下为例进行安装过程说明，其他aarch64、mips64等平台安装过程一致，安装过程中选择对应平台的产品包介质进行安装即可，不再做重复说明。

AMP监控平台的产品安装包括web控制台、基础设施监控服务、运维工单系统、容器云监控服务的安装，其中容器云监控服务为可选安装。

第四章介绍了web控制台的安装过程，安装AMP产品必须安装。

第五章介绍了aalarm告警服务器的安装过程，安装AMP产品必须安装

第六章介绍了基础设施监控服务的安装过程，安装AMP产品必须安装。

第七章介绍了运维工单系统的安装过程，安装AMP产品必须安装。

第八章介绍了容器云监控服务的安装过程，安装AMP产品可选安装。

第九章介绍了AMP高可用部署架构的安装说明。

第十章介绍了AMP产品运行所需的相关第三方组件及数据库的安装过程参考说明。

12 第4章 安装Web控制台

Web控制台是AMP产品的核心组件，提供统一的用户、权限管理以及平台服务接入管理功能。

13 4.1 产品介质说明

Web控制台组件相关安装介质如下：

组件名称	文件名	说明
Web控制台	amp-console-v3.3.tar.gz	Web控制台组件，springboot应用

表4-1 Web控制台产品介质

注意：安装后运行web控制台需要提供license.xml文件，请向您的产品服务提供商获取。

14 4.2 安装Web控制台

15 4.2.1 安装准备

 JDK

应用运行需要JDK8 环境，参考附录10.1 安装说明

 Redis

应用运行需要Redis缓存服务，参考附录 10.4 安装说明。

 MySQL数据库

系统默认推荐使用MySQL，参考附录10.2节MySQL的安装说明，如已经安装请跳过。如果您使用其他类型的数据库，请参考对应厂商说明帮助手册进行安装。

16 4.2.2 安装说明

创建AMP产品安装根目录，指定${PATH} 为实际路径，将amp-console-v3.3.tar.gz解压到对应目录及完成产品包安装，/${PATH}/AMP/amp-console为产品解压后的目录。

```
# mkdir -p /${PATH}/AMP
# tar -zxvf amp-console-v3.3.tar.gz -C /${PATH}/AMP
```

如上，即完成Web控制台应用的解压工作，接下来修改相关参数配置。

17 4.2.1 配置参数

修改amp-console/conf/application.yml文件，该配置文件为SpringBoot应用的默认配置文件，active的值为prod，其对应生效的文件是application-prod.yml,采用的是MySQL数据库连接配置。 系统中提供如下可选的配置文件:

```
    文件名          说明
    application-dev.yml H2数据库作为持数据持久化存储的配置，开发环境阶段使用，生产环境不建议使用
    application-prod.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-samle-dm.yml     达梦数据库作为数据持久化存储的配置
    application-sample-gbase8s.yml       南大通用Gbase8s作为数据持久化存储的配置
    application-sample-kingbasees.yml    人大进仓KingbaseES作为数据持久化存储的配置
    application-sample-shentong.yml      神舟通用数据库作为数据持久化存储的配置
```

表格4-2 Web控制台应用配置文件

用户可根据实际部署环境修改application.yml文件中的active值为prod、sample-kingbasees、sample-dm、sample-shentong、sample-gbase-8s来切换不同环境的配置。 下面以采用MySQL配置的application-prod.yml文件为例，说明相关主要参数配置。

 server.port 参数指定了该web应用的默认端口，默认值为9000

 spring.redis 指定了应用连接Redis相关配置，需要根据实际部署环境进行修改。

- timeout 为超时时间，默认3600s
- host为redis的ip地址，默认值localhost
- port为redis端口，默认值6379
- password 为redis连接密码

 spring.datasource 为数据库连接配置， 需要根据实际部署环境进行修改。

- url为数据库JDBC连接配置，包含数据库地址、端口、数据库名称等参数
- Username 指定数据库连接用户名
- Password 指定数据库连接密码

配置参考样例如下:

```
    redis:
      timeout: 3600
      host: localhost
      port: 6379
      password: root
    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
      <省略其他>
```

18 4.2.2 初始化数据库

下面以mysql数据库的初始化为例进行说明。安装完成后登录mysql数据库。

```
mysql -uusername -ppassword
```

amp-console/sql/mysql目录存放create.sql数据库创建脚本文件，initial.sql数据库初始化脚本文件，执行SQL脚本进行数据初始化。 create database amp_console; use amp_console; source /${PATH}/AMP/amp-console/sql/mysql/create.sql; source /${PATH}/AMP/amp-console/sql/mysql/initial.sql;

初始化完成后输入exit退出mysql数据库。

19 4.3 安装后的工作

20 4.3.3 了解产品目录结构

目录	说明
bin	控制台组件的启动脚本。
boot	控制台程序的jar文件。
conf	一些配置文件。
lib	应用程序依赖的一些jar包。
sql	控制台对应的amp_console数据库多种版本的sql创建及初始化脚本文件。
HELP.md	帮助文档，对控制台项目的补充说明。

表格4-3 Web控制台目录结构

21 4.3.2 启动运行服务

修改完amp-console的配置文件后，后台启动Web控制台。

```
nohup /${PATH}/AMP/amp-console/bin/startup.sh &
```

查看Web控制台运行状态，若端口9000存在，表示启动成功。 访问浏览器验证：http://amp-console_ip:9000,出现如下图所示登录页面。

```
netstat -lntp | grep 9000
```

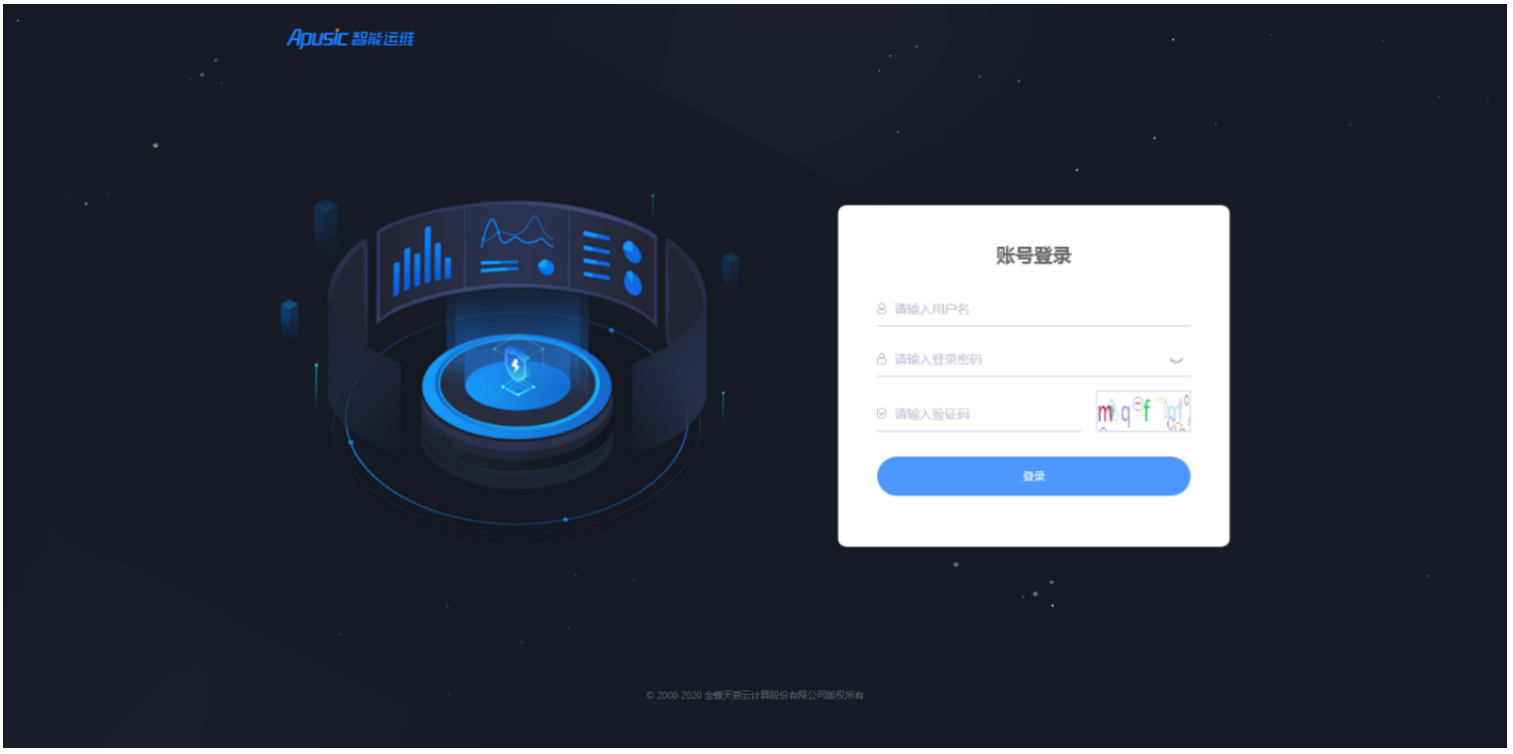


图4-1平台登录页

输入用户名：admin和默认 密码：Admin$123456，登录成功，则表明Web控制台部署成功。

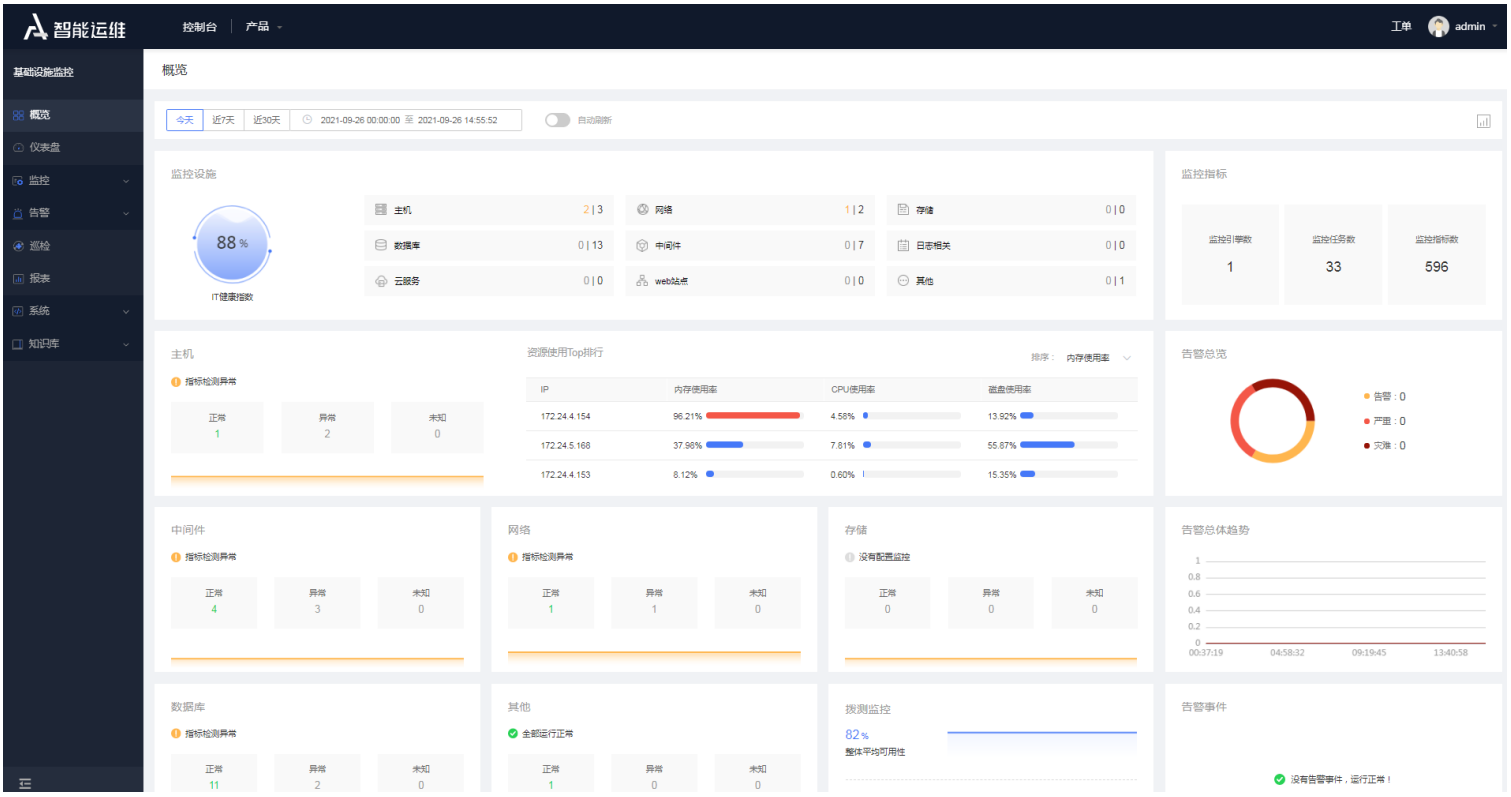


图4- 2 Web控制台首页

22 4.3.3 停止运行服务

目前可以根据端口号查找出该应用程序的进程，使用kill命令终止Web控制台进程。

```
# netstat -lntp | grep 9000
tcp6       0      0 :::9000       :::*            LISTEN      19358/java
# kill -9 19358
```

23 4.3.4 卸载服务

执行下列命令，删除Web控制台部署目录，即可完成卸载操作。

```
rm -rf /${PATH}/AMP/amp-console*
```

24 第5章 安装Apusic Alarm平台

25 5.1 产品介质说明

组件名称	文件名	说明
AAlarm	alarm-manager-SE-v1.0.tar.gz	智能告警平台，SprinBoot应用
Kafka	kafka_2.12-2.7.0.tgz	消息中间件，建议不与其他应用共用。

表4-1 Apusic Alarm产品相关安装包说明表

26 5.2 安装Apusic Alarm平台

27 5.2.1 安装说明

 JDK

应用运行需要JDK8 环境，参考附录10.1 安装说明

 Redis

应用运行需要Redis缓存服务，参考附录 10.4 安装说明。

 MySQL数据库

系统默认推荐使用MySQL，参考附录10.2节MySQL的安装说明，如已经安装请跳过。如果您使用其他类型的数据库，请参考对应厂商说明帮助手册进行安装。

28 5.2.2 安装准备

本手册以在Linux x86_64环境下为例进行安装过程说明，其他aarch64、mips64等平台安装过程一致，安装过程中选择对应平台的产品包介质进行安装即可，不做重复说明。

根据业务需要监控应用较多，建议用两台以上部署

创建Apusic Alarm产品安装根目录，**指定${PATH} 为实际路径**。Apusic Alarm产品包里的所有组件建议都安装在/${PATH}/AAlarm目录下

```
mkdir -p /${PATH}/AAlarm
```

29 5.2.3 安装kafka

1. 解压安装包到/${PATH}/AOPS/目录下

```
tar -zxvf kafka_2.12-2.7.0.tgz -C  /${PATH}/AOPS
```

2. 创建kafka日志目录与zookeeper存储快照的目录

```
#创建kafka日志目录
mkdir /${PATH}/AOPS/kafka_2.12-2.7.0/log
#创建zookeeper存储快照的目录
mkdir /${PATH}/AOPS/kafka_2.12-2.7.0/zookeeper
```

3. 修改配置参数

   - 进入kafka配置文件目录

   ```
   cd /${PATH}/AOPS/kafka_2.12-2.7.0/config
   ```

   - 修改 server.properties

   ```
   vim server.properties

   ###修改为kafka实际部署IP
   listeners=PLAINTEXT://:9092
   advertised.listeners=PLAINTEXT://localhost:9092
   zookeeper.connect=localhost2181
   ###修改为kafka日志文件到对应目录
   log.dirs=/${PATH}/AOPS/kafka_2.12-2.7.0/log
   ```

   - 修改producer.properties

   ```
   vim producer.properties

   ###修改为kafka实际部署IP
   bootstrap.servers=localhost:9092
   ```

   - 修改zookeeper.properties

   ```
   vim zookeeper.properties

   ###修改为zookeeper对应目录
   dataDir=/${PATH}/AOPS/kafka_2.12-2.7.0/zookeeper
   ```

30 5.2.4 配置参数

修改aalarm-manager_SE/conf/application.yml文件，该配置文件为SpringBoot应用的默认配置文件，active的值为prod，其对应生效的文件是application-prod.yml,采用的是MySQL数据库连接配置。 系统中提供如下可选的配置文件:

```
    文件名          说明
    application-dev.yml H2数据库作为持数据持久化存储的配置，开发环境阶段使用，生产环境不建议使用
    application-prod.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-samle-dm.yml     达梦数据库作为数据持久化存储的配置
    application-sample-gbase8s.yml       南大通用Gbase8s作为数据持久化存储的配置
    application-sample-kingbasees.yml    人大进仓KingbaseES作为数据持久化存储的配置
    application-sample-shentong.yml      神舟通用数据库作为数据持久化存储的配置
```

表格4-2 Web控制台应用配置文件

用户可根据实际部署环境修改application.yml文件中的active值为prod、sample-kingbasees、sample-dm、sample-shentong、sample-gbase-8s来切换不同环境的配置。 下面以采用MySQL配置的application-prod.yml文件为例，说明相关主要参数配置。

 server.port 参数指定了该web应用的默认端口，默认值为9000

 spring.redis 指定了应用连接Redis相关配置，需要根据实际部署环境进行修改。

- timeout 为超时时间，默认3600s
- host为redis的ip地址，默认值localhost
- port为redis端口，默认值6379
- password 为redis连接密码

 spring.datasource 为数据库连接配置， 需要根据实际部署环境进行修改。

- url为数据库JDBC连接配置，包含数据库地址、端口、数据库名称等参数
- Username 指定数据库连接用户名
- Password 指定数据库连接密码

配置参考样例如下:

```
    redis:
      timeout: 3600
      host: localhost
      port: 6379
      password: root
    datasource:
    type: com.zaxxer.hikari.HikariDataSource
      url: jdbc:mysql://localhost:3306/aalarm?
useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC
      username: root
      password: root
      <省略其他>
```

31 5.2.5 初始化数据库

下面以mysql数据库的初始化为例进行说明。安装完成后登录mysql数据库。

```
mysql -uusername -ppassword
```

amp-console/sql/mysql目录存放create.sql数据库创建脚本文件，initial.sql数据库初始化脚本文件，执行SQL脚本进行数据初始化。

```
create database aalarm;
use aalarm;
source /${PATH}/AMP/aalarm/sql/mysql/create.sql;
source /${PATH}/AMP/aalarm/sql/mysql/initial.sql;
```

初始化完成后输入exit退出mysql数据库。

32 5.3 安装后的工作

33 5.3.3 了解产品目录结构

目录	说明
bin	控制台组件的启动脚本。
boot	控制台程序的jar文件。
conf	一些配置文件。
lib	应用程序依赖的一些jar包。
sql	控制台对应的aalarm数据库多种版本的sql创建及初始化脚本文件。
HELP.md	帮助文档，对控制台项目的补充说明。

表格4-3 Web控制台目录结构

34 5.3.2 启动运行服务

修改完aalarm的配置文件后，后台启动Web控制台。

```
nohup /${PATH}/AALARM/aalarm-manager_SE/bin/startup.sh &
```

若端口9016存在，表示启动成功。

```
netstat -ntpl | grep 9016
```

35 第6章 安装基础设施监控服务

基础设施监控服务作为AMP平台产品的核心服务之一，提供IT基础设施的统一监控管理功能。本章介绍基础设施监控服务及相关组件的安装说明。

36 6.1 产品介质说明

基础设施监控服务产品相关安装介质文件：

组件名称	文件名	说明
Web监控平台	amp-infra-monitor-v3.3.tar.gz	web监控平台组件，springboot应用
告警服务器	aalarm-manager_SE-v1.1.tar.gz	Prometheus监控引擎的报警服务器组件，用于监控报警信息的通知
融合组件	amp-components.linux-amd64.zip	融合了prometheus监控引擎，prom-agent监控服务器代理，nettopo以及blackbox四个组件

表5-1 基础设施监控服务相关产品组件

注意：安装后运行基础设施监控服务web 监控平台需要提供license.xml文件，请向您的产品服务提供商获取。

37 6.2 安装基础设施监控服务

38 6.2.1 安装监控融合组件

此组件融合了prometheus监控引擎，prom-agent监控服务器代理，nettopo以及blackbox四个组件，各部分功能如下

- 监控服务器prom agent由Prometheus监控引擎和监控服务器代理两部分组成，基础设施监控服务的web监控平台通过监控服务器代理和Prometheus引擎进行交互通信实现监控任务和告警规则的创建和配置管理。

39 6.2.1.1 安装说明

创建AMP产品安装根目录，将amp-components.linux-amd64.zip解压到对应目录及完成产品包安装。注意：${PATH}为部署环境实际目录

```
# unzip amp-components.linux-amd64.zip -C /${PATH}/AMP
```

40 6.2.1.2 配置参数

1. 编辑/${PATH}/AMP/amp-component/runtime/config/prom_agent_standalone.conf配置信息

- ☐ 修改agent_id为5.3.3.2章节注册时的引擎ID，默认值为1。注意：若平台只有一个监控服务器则无需修改保持默认值即可。
- ☐ 修改监控服务平台url 地址，将amp_monitoring_url的值修改为amp-monitoring部署的ip。
- ☐ blackbox_export_url修改为web站点部署的ip地址

```
#AMP平台注册的引擎id
agent_id=1

#Web监控平台地址
amp_monitoring_url = http://localhost:9002

#站点监控blackbox地址
blackbox_export_url = localhost:9115
```

2. 编辑/${PATH}/AMP/amp-component/config.yaml，修改redis部署所在的ip和端口，以及【监控服务器】部署所在的ip地址

```
redis:
  enable: true
  nodes: localhost:6379
  #password: password
  #db: 0
  #nodes: 172.24.4.111:28001,172.24.4.111:28002,172.24.4.111:28003
  #master: mymaster
prometheus:
  url: http://localhost:9090
```

41 6.2.2 安装Web监控平台

42 6.2.2.1 安装准备

参考4.2.1节Web控制台的安装准备说明，如果已经安装相关依赖环境组件则可跳过本步骤。

43 6.2.2.2 安装说明

创建AMP产品安装根目录，指定${PATH} 为实际路径，将amp-infra-monitor-v3.3.tar.gz解压到对应目录及完成产品包安装，/${PATH}/AMP/amp-infra-monitor为产品解压后的目录。

```
# mkdir -p /${PATH}/AMP
# tar -zxvf amp-infra-monitor-v3.3.tar.gz -C /${PATH}/AMP
```

将licensex.xml授权文件拷贝到/${PATH}/AMP/amp-infra-monitor根目录，否则监控平台web应用将无法正常启动。 如上即完成Web监控平台应用的解压工作，接下来修改相关参数配置。 5.2.6.3 配置参数 修改amp-infra-monitor/conf/application.yml文件，该配置文件为SpringBoot应用的默认配置文件，active的值为prod，其对应生效的文件是application-prod.yml,采用的是MySQL数据库连接配置。 系统中提供如下可选的配置文件:

文件名	说明
application-dev.yml	H2数据库作为持数据持久化存储的配置，开发环境阶段使用，生产环境不建议使用
application-prod.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-samle-dm.yml	达梦数据库作为数据持久化存储的配置
application-sample-gbase8s.yml	南大通用Gbase8s作为数据持久化存储的配置
application-sample-kingbasees.yml	人大进仓KingbaseES作为数据持久化存储的配置
application-sample-shentong.yml	神舟通用数据库作为数据持久化存储的配置

表5-1 基础设施监控服务Web监控平台应用配置文件

用户可根据实际部署环境修改application.yml文件中的active值为prod、sample-kingbasees、sample-dm、sample-shentong、sample-gbase-8s来切换不同环境的配置。 下面以采用MySQL配置的application-prod.yml文件为例，说明相关主要参数配置。

```
Vi /${PATH}/AMP/amp-infra-monitor/conf/application-prod.yml
```

 server.port 参数指定了该Web监控平台的默认端口，默认值为9002 spring.redis 指定了应用连接Redis相关配置，需要根据实际部署环境进行修改。

- timeout 为超时时间，默认3600s
- host为redis的ip地址，默认值localhost
- port为redis端口，默认值6379
- password 为redis连接密码 spring.datasource 为数据库连接配置， 需要根据实际部署环境进行修改。
- url为数据库JDBC连接配置，包含数据库地址、端口、数据库名称等参数
- Username 指定数据库连接用户名
- Password 指定数据库连接密码 amp为相关连接配置信息，需要根据实际部署环境进行修改。
- amp.consol.url 为web控制台 amp-console应用的部署地址，默认http://localhost:9000
- amp.agent.url 为监控服务器agent的部署地址，默认值http://localhost:8100，修改为对应的地址。
- amp.alarm.url 为告警服务器代理alarm-gateway的地址，默认值http://localhost:8200，修改为实际部署地址。
- amp.net_topo.url为网络拓扑组件amp-nettopo的地址，默认值为http://localhost:9808，修改为实际部署地址。
- amp.workorder.url为运维工单系统amp-workorder的地址，默认值为http://localhost:9014，修改为实际部署地址。 配置参考样例如下:

```
redis:
  timeout: 3600
  host: localhost
  port: 6379
  password: root
datasource:
```

```
    type: com.zaxxer.hikari.HikariDataSource
    url: jdbc:mysql://localhost:3306/amp_monitoring
  ?
useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=U
    username: root
    password: root
  amp:
    console:
      url: http://localhost:9000
    agent:
      url: http://localhost:8100
  alarm:
      url: http://localhost:8200
    net_topo:
      url: http://localhost:9808
  <省略其他>
```

注意：没有授权文件监控平台web应用将无法启动，请联系您的销售或者技术顾问获取授权文件。

44 6.2.2.3 初始化数据库

登录mysql数据库，执行amp-infra-monitor/sql/mysql目录下的create.sql数据库创建脚本文件，initial.sql数据库初始化脚本文件，初始化web基础设施监控服务的数据库。

```
create database amp_monitoring;
use amp_monitoring;
source /${PATH}/AMP/amp-infra-monitor/sql/mysql/create.sql;
source /${PATH}/AMP/amp-infra-monitor/sql/mysql/initial.sql;
```

初始化完成后输入exit退出mysql数据库。

45 6.3 安装后的工作

46 6.3.1 了解产品目录结构

47 6.3.1.1 Web监控平台

目录	说明
bin	监控平台组件的启动脚本。
generator	生成snmp监控模板yml配置的工具方法。
share_config	监控平台下Web站点监控，采集器，snmp采集模板的信息。
conf	一些配置文件。
lib	应用程序依赖的一些jar包。
sql	监控平台对应的amp_monitoring数据库多种版本的sql创建及初始化脚本文件。
HELP.md	帮助文档，对监控平台项目的补充说明。

表5-6 Web监控平台目录结构

48 6.3.1.2 融合组件

目录	说明
components	组件启动脚本
prometheus	集成了prometheus监控引擎
runtime/config	保存监控服务器程序的配置文件
config.yaml	网络拓扑组件配置文件

49 6.3.2 启动运行服务

50 6.3.2.1 启动融合组件

nohup ./components & （启动后会占用三个端口，9808默认为为prom-agent和nettopo服务端口， 9115默认为blackbox服务端口，9090默认为prometheus服务端口）

若端口9090,9808,9115都存在，表示启动成功

```
netstat -lntp | grep 9808
netstat -lntp | grep 9090
netstat -lntp | grep 9115
```

说明：若需要修改以上三个端口，则需要在启动命令添加一下参数

- --listen : 修改prom-agent和nettopo服务端口，示例：

  ```
  nohup  ./components --listen=0.0.0.0:9809 &
  ```

- --prome_listen：修改prometheus服务端口，示例：

  ```
  nohup  ./components --prome_listen=0.0.0.0:9091 &
  ```

  需要提前修改prom_agent_standalone.conf配置文件中的prome_reload_url参数

- --blackbox_listen：修改blackbox端口，示例：

  ```
  nohup  ./components --blackbox_listen=0.0.0.0:9116 &
  ```

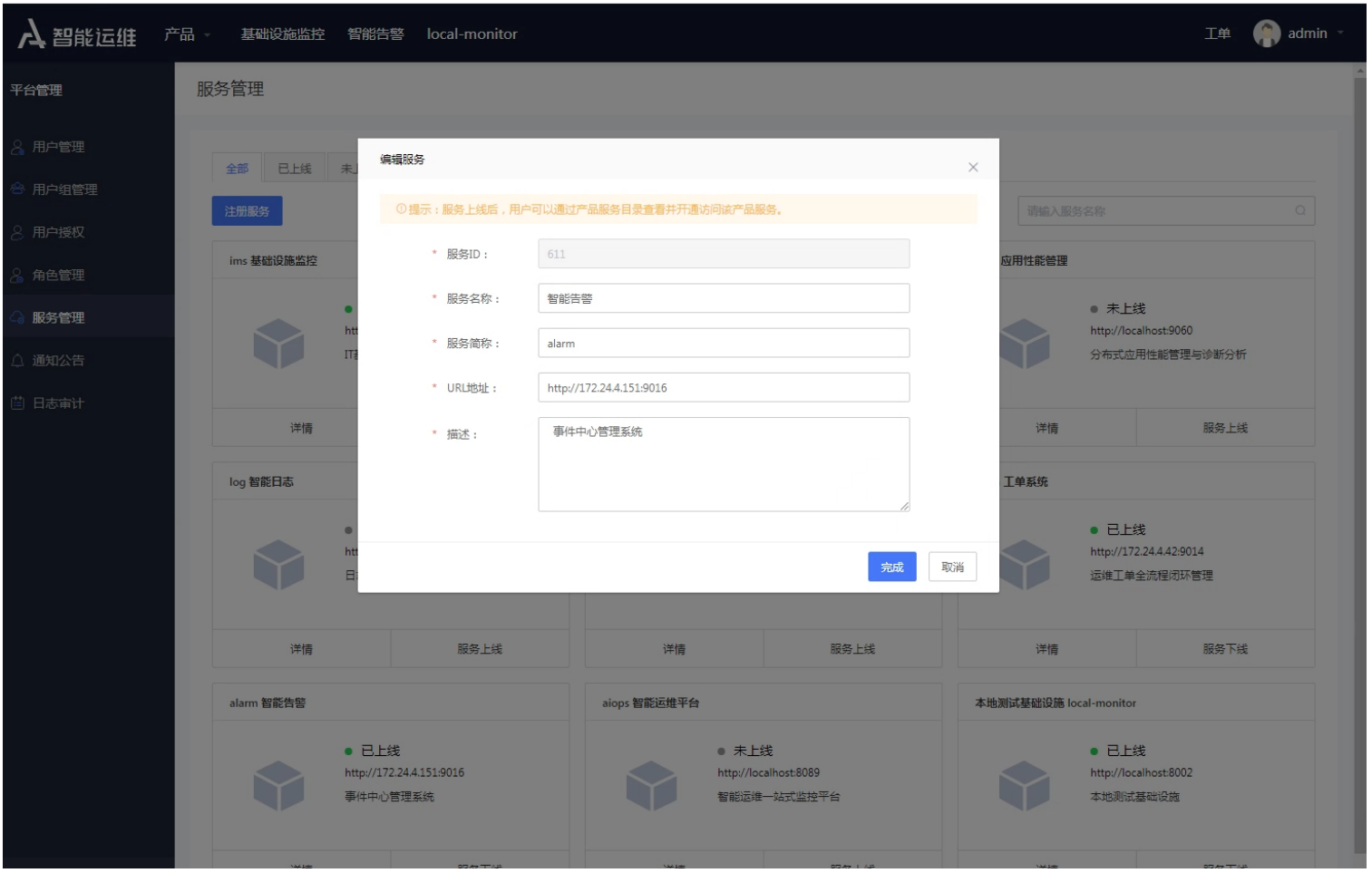
  需要提前修改prom_agent_standalone.conf配置文件中的blackbox_export_url参数

51 6.3.3 注册为平台服务

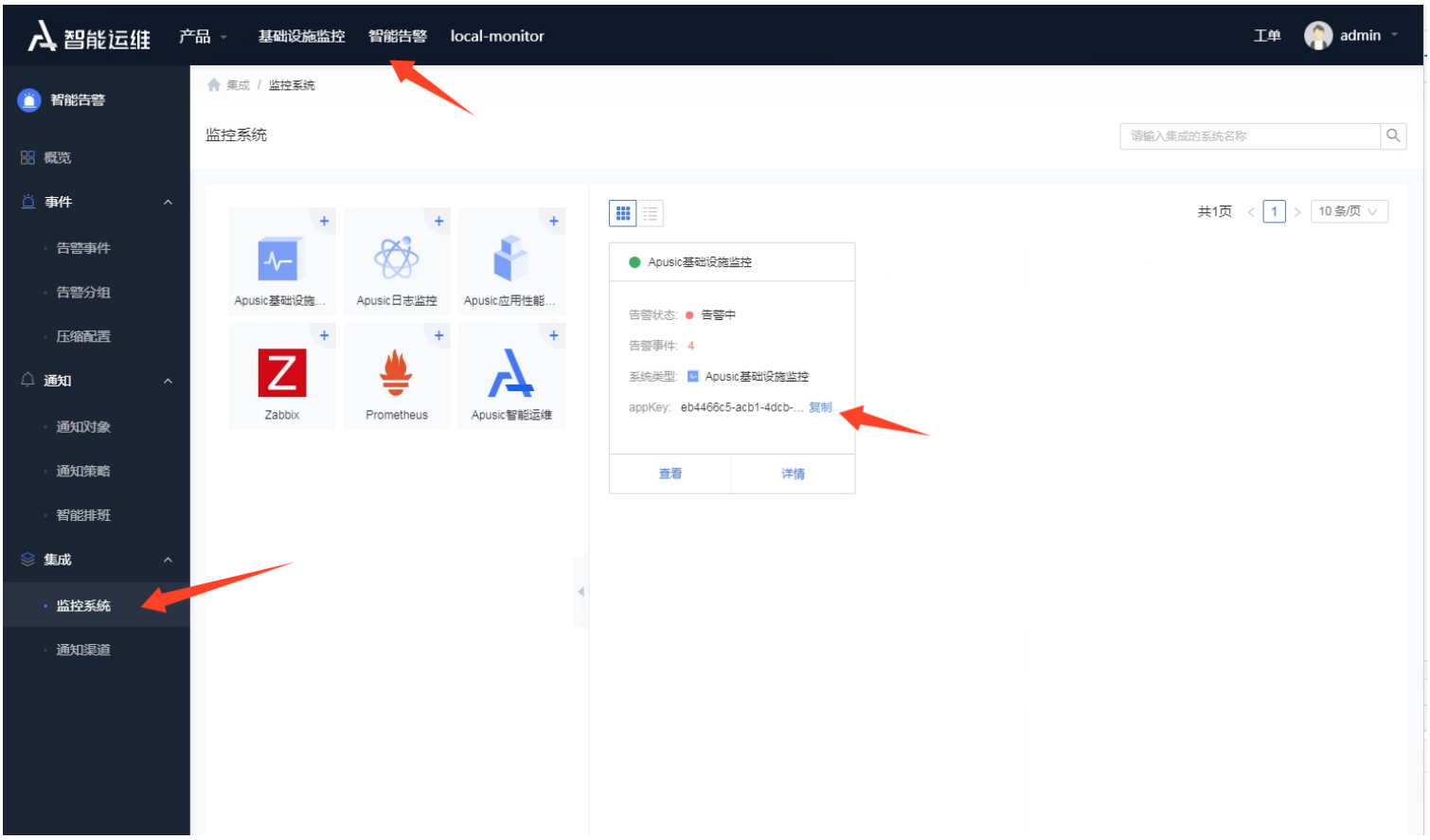
基础设施监控服务部署完成并注册为平台服务后，用户才可以访问使用。

52 6.3.3.1 AAlarm注册为平台服务

登录访问Web控制台，选择【平台管理】->【服务管理】。编辑【智能告警服务】，服务URL址。修改完成，点击服务上线，完成告警服务的注册上线。



进入智能告警系统，在监控系统添加Apusic基础设施监控，复制appKey,最后启动基础设施监控时要用。


智能运维
产品
基础设施监控
智能告警
local-monitor
工单
admin
智能告警
概览
事件
告警事件
告警分组
压缩配置
通知
通知对象
通知策略
智能排班
集成
监控系统
通知渠道
集成 / 监控系统
监控系统
请输入集成的系统名称
Apusic基础设施...
Apusic日志监控
Apusic应用性能...
Zabbix
Prometheus
Apusic智能运维
共1页
1
10 条/页
Apusic基础设施监控
告警状态: 告警中
告警事件: 4
系统类型: Apusic基础设施监控
appKey: eb4466c5-acb1-4dcb-... 复制
查看
详情

53 6.3.2.4 启动web监控平台

修改配置文件，使用上一步复制的appkey替换/${PATH}/AMP/amp-infra-monitor/con/applicaton-prod的appkey,保存后退出。

```
  connectTimeout: 10000
  #从连接池中获取到连接的最长时间
  connectionRequestTimeout: 500
  #数据传输的最长时间
  socketTimeout: 10000
  #提交请求前测试连接是否可用
  staleConnectionCheckEnabled: true

amp:
  console:
    url: http://172.24.4.156:9000
  monitor:
    url: http://172.24.4.42:9002
    report:
      record.generate.day.cron: 1 0 0 * * ?
      record.generate.week.cron: 1 0 0 ? * MON
      record.generate.month.cron: 1 0 0 1 * ?
    asset:
      record.generate.day.cron: 1 0 0 * * ?
      wo.record.generate.day.cron: 1 0 0 * * ?
      file.clear.day.cron: 1 0 0 * * ?
    promServer:
      prom.update.runningstatus.second.cron: 0/30 * * * * ?
  agent:
    url: http://172.24.4.42:8100
  net_topo:
    url: http://172.24.4.42:9808
  workorder:
    url: http://172.24.4.42:9014

alarmmanager:
  url: http://172.24.4.151:9016
  appKey: eb4466c5-acb1-4dcb-a12c-8f5e02970492

cmdb:
  type: apusic
  url: http://172.24.4.163:9020

adpr:
  integration: false
  app:
    url: http://172.24.4.163:9527

# 一些开发配置
dev:
  gbase:
    # 是否生成col_tpl表sample_data的数据文件和update.sql脚本，给gbase 8s数据库使用
    genSampleData: false

rsa:
  publicKey: MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCpXCNp7kkNArJxlDNclZwdQ0q85St4vIRNh1QRMJJlsjhV9uY6ykRMUJrUd64ieFvt3tm2fiAcy4Qr6UQ3CqGdL8QJdMD0qKb7bsysw6r9KvEKP6Vd9NF/CcuT6HerJjW4DazRxooByhKzxJ
RZP+h8m/ah7XDtMILyxxaqfQIDAQAB
  privateKey: MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAKlcI2nuSQ0CsnGUM1yVnB1DSrzlK3i8hE2HVBEwkmWyOFX25jrKRExQmtR3riJ4W+3e2bZ+IBzLhCvpRDcKoZ0vxAl0wPSopvtuzKzDqv0q8Qo/pV300X8Jy5Pod6smNbgNr
gHKErPEn+akVFk/6Hyb9qHtcO0wgvLHFqp9AgMBAAECgYADzc390xxQ7n6TjoCY08Nt5NUI6RzQ2WsfUqrQuAaRwi8Bv0nsiB6AmE7h9s4yeAGYlijNsIeIHgoZ8mNV2SdIFymtiFgesfXMdNmlaJxwhkVr0OVzKaPwqgFdxcmUciqF+126yj7KGINJXfjFADN
AKji4Kfew1df8hHoAQJBAN/vA+HJa0f368J7cEmwnXAf7d2NMP5Us4WRqDRD4F81EnnslTrQcVr5rBAL6hNY1gyUWy9Xwd6g8QTz0fp95dkCQQDBnI2R++CrCGFMpaOOzJEG2BruZTVFCIyyLYi6WzL2e8xuQzhDZPy37lRPh16sImxkNftHxTdxHf/JGa/aMA9
AgrFzmY49f6ZRQZNbaJacfDkWUWy7AGSc1cKiiV5oEAjgfKo4Ojg3QDphUogxmBML8hGP2mSkcHOL1zEaBtD9mQJAH+ZZdQoZFFUHIwfPkQGni4cjuM+4HS2xcNILyxinpvDy19//bys+cFe2UHwvXxE05rX2NvcJkXut30Z6MOrmOQJBANbQgY1rO7wLMNGexf
NGDcqVBs5gm/r9xo6cJ7+ASrQiKwpSZ/GqEGmlDkjeQ6FxMXdmlQz5LQXlIlhgfGo=
```

执行启动web监控平台命令。

```
nohup /${PATH}/AMP/amp-infra-monitor/bin/startup.sh &
```

54 6.3.3.1 基础设施监控注册为平台服务

登录访问Web控制台，选择【平台管理】->【服务管理】。编辑【基础设施监控服务】，服务URL址。修改完成，点击服务上线，完成基础设施监控服务的注册上线。

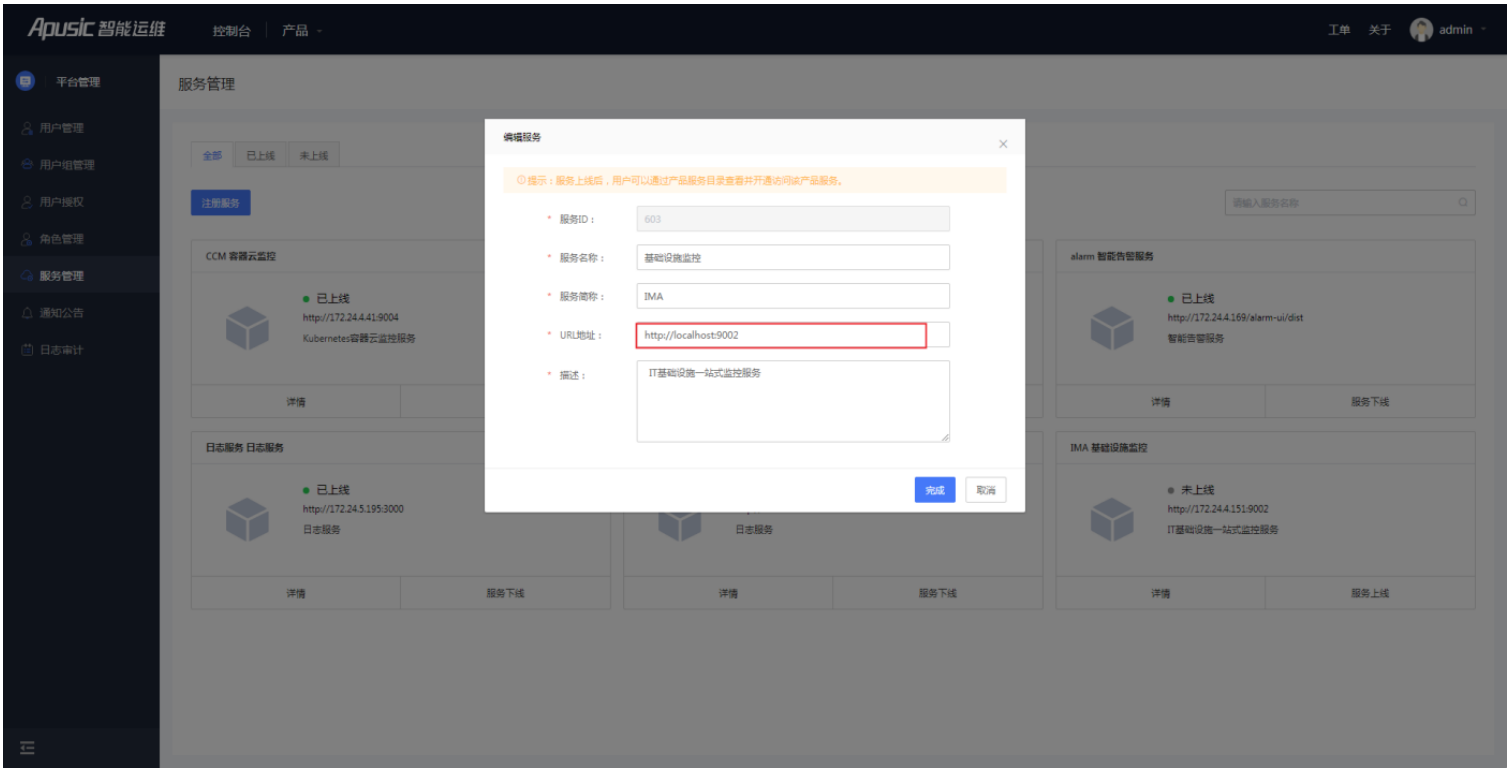


图5-4 注册基础设施监控服务

1. 点击控制台产品下拉列表，切换至【基础设施监控】服务，出现如下图则表明注册服务成功 。

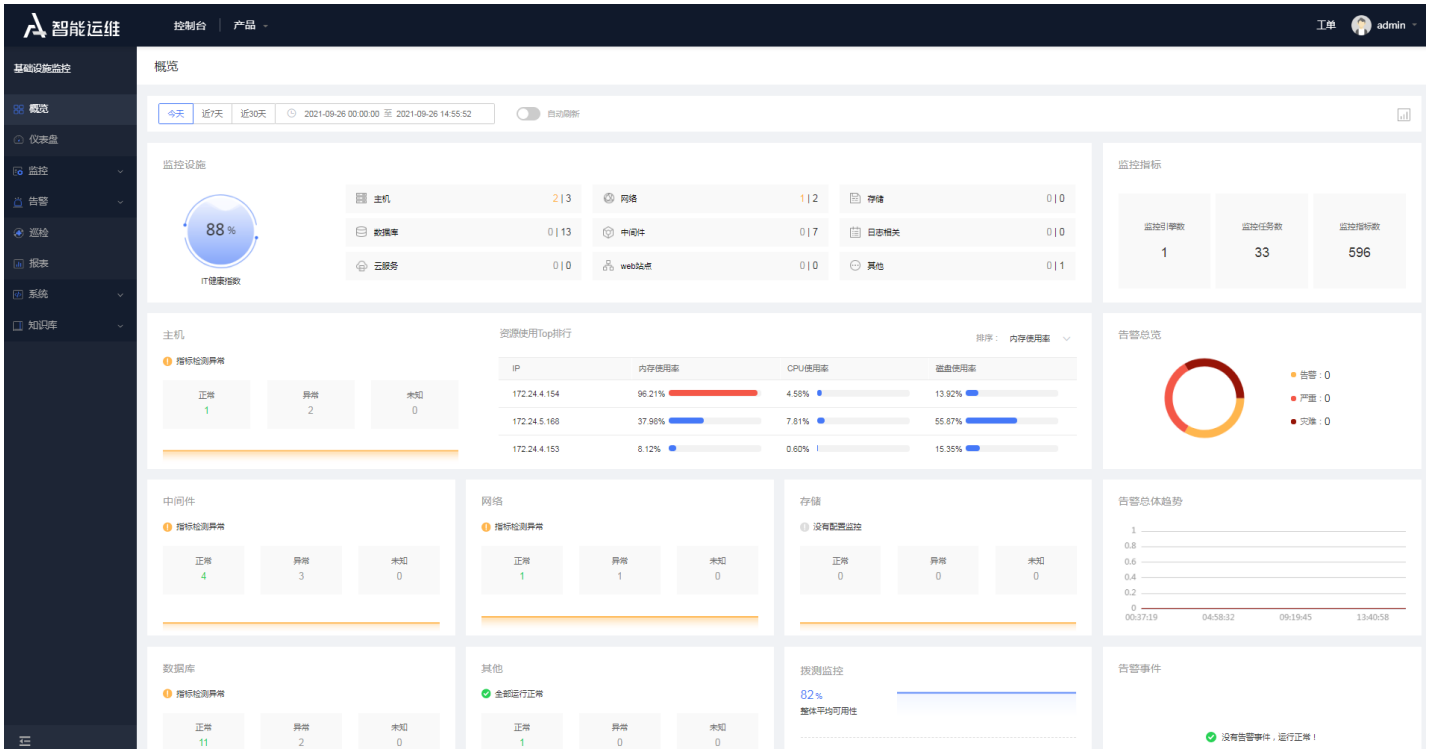


图5-5 基础设施监控服务首页

55 6.3.3.2 添加数据中心

执行下面sql脚本，新增数据中心，例如：新增加的数据中心id为：2， 数据中心名称为：深圳数据中心（如果amp系统与aump系统同时部署，该数据中心id，数据中心名称需要与aump系统中保持一致）

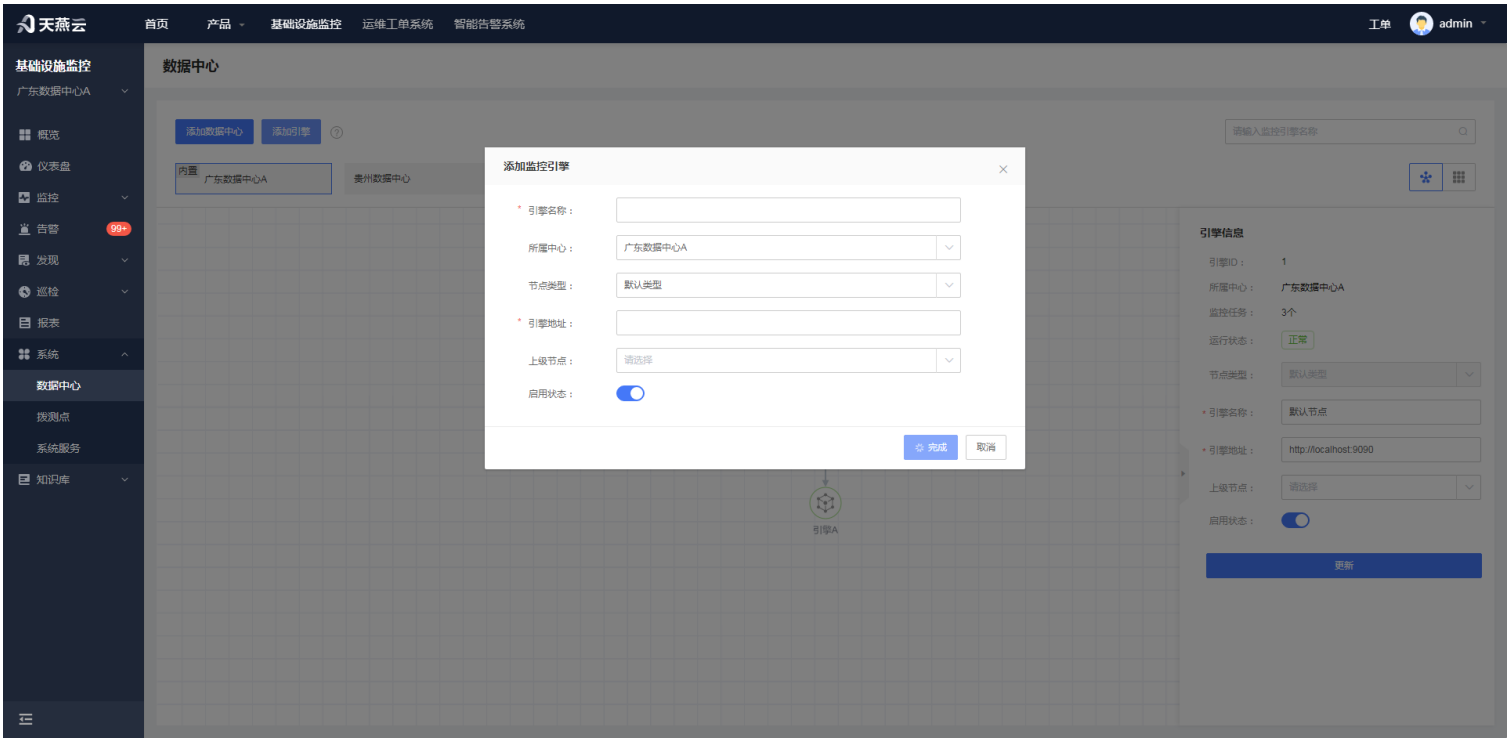
```
INSERT INTO `amp_monitoring`.`data_center`(`id`, `created_by`, `created_date`, `last_modified_by`, `last_modified_date`, `name`, `category`) VALUES (2, 'admin', '2022-08-29 13:50:35', 'admin', '2022-08-29 13:50:35', '深圳数据中心', 'OFFICAL');
```

注意：每个数据中心下至少有一个监控引擎。

56 6.3.3.3 添加监控引擎

56.1 1、通过页面

进入监控引擎管理界面，点击添加监控引擎，进行添加监控引擎操作，如下图所示。



监控引擎名称：用于标识监控引擎的名称，必填项。

节点类型：包括三种：默认类型，容器监控类型，日志监控类型，默认类型用于创建基础类型监控任务，容器监控类型用于创建k8s容器监控任务，日志监控类型用于日志性监控任务。根据实际监控的需求设置相应的节点类型，必填项。

引擎地址：访问prometheus的地址，或者是在prometheus高可用部署时，使用的nginx的代理访问地址，必填项。

引擎创建成功后，会分配一个引擎ID，将引擎ID配置到prom-agent的agent_id。

56.2 2、通过脚本

执行sql脚本，新增监控引擎，例如：新增加的数据中心id为：2，监控引擎id为：2，数据中心id为：2（表示将该引擎关联到Id=2的数据中心下），监控引擎名称为：深圳监控节点，修改监控引擎地址"http://127.0.0.1:9090"为真实的部署地址

```
INSERT INTO `amp_monitoring`.`prom_server`(`id`, `created_by`, `created_date`, `last_modified_by`,
`last_modified_date`, `federate_cluster_mode`, `job_type`, `name`, `parent_node`, `role`,
`running_status`, `status`, `type`, `url`, `weight`, `data_center_id`, `category`) VALUES (2,
'admin', '2019-11-13 17:01:34', 'admin', '2022-12-26 06:56:57', 'CLOSE', 'STATIC', '深圳监控节点',
NULL, 'MASTER', 'ON_LINE', 'ENABLE', 'TYPE_PROME_DEFAULT', 'http://127.0.0.1:9090', 1, 2,
'OFFICAL');
```

57 6.3.4 停止运行服务

58 6.3.4.1 关闭监控服务器

查看其端口8100所运行的进程的pid，使用kill命令终止监控服务器进程。

```
# netstat -lntp | grep 8100
tcp6       0      0 :::8100       :::*            LISTEN      19358/java
# kill -9 19358
```

查看其端口9090所运行的进程的pid，使用kill命令终止prometheus进程。

```
# netstat -lntp | grep 9090
tcp6       0      0 :::9090       :::*            LISTEN      19358/java
# kill -9 19358
```

59 6.3.4.2 关闭网络拓扑组件

查看其端口9093所运行的进程的pid，使用kill命令终止网络拓扑进程。

```
# netstat -lntp | grep 9808
tcp6       0      0 :::9808       :::*            LISTEN      19358/java
# kill -9 19358
```

60 6.3.4.3 关闭站点监控组件

查看其端口9115所运行的进程的pid，使用kill命令终止Web站点监控服务进程。

```
# netstat -lntp | grep 9115
tcp6       0      0 :::99115        :::*            LISTEN      19358/java
# kill -9 19358
```

61 6.3.4.4 关闭Web监控平台

查看端口9002所运行的进程的pid，使用kill命令终止Web监控平台进程。

```
# netstat -lntp | grep 9002
tcp6       0      0 :::9002       :::*            LISTEN      19358/java
# kill -9 19358
```

62 6.3.4.5 关闭融合组件

查看9090,9155或者9808任一端口所运行的进程的pid，使用kill命令终止

```
# netstat -lntp | grep 9090
tcp6       0      0 :::9090       :::*            LISTEN      19358/java
# kill -9 19358
```

63 6.3.5 卸载服务

64 6.3.5.1 卸载单独服务

1. 执行下列命令，删除监控服务器安装部署目录。

   rm –rf /${PATH}/AMP/prom-agent

2. 执行下列命令，删除服务器安装部署目录。

   rm –rf /${PATH}/AALARM/alarm-manager*

3. 执行下列命令，删除告警服务器代理安装部署目录。

   rm –rf /${PATH}/AMP/ prom-agent*

4. 执行下列命令，删除网络拓扑安装部署目录。

   rm –rf /${PATH}/AMP/amp-nettopo-*

5. 执行下列命令，删除站点监控安装部署目录。

   rm –rf /${PATH}/AMP/monitor-blackbox*

6. 执行下列命令，删除Web监控平台安装部署目录，即可完成卸载操作。

   rm –rf /${PATH}/AMP/ amp-infra-monitor*

65 6.3.5.2 卸载全部服务

执行下列命令，删除Web监控平台安装根目录，即可完成卸载操作。

```
rm -rf /${PATH}/AMP
```

66 第7章 安装运维工单系统

运维工单系统作为AMP平台产品的基础支撑服务，提供工单的提单及工单的闭环流程管理功能，是整个运维平台的核心关键系统之一。 本章介绍运维工单系统的安装过程及配置说明。

67 7.1 产品介质说明

运维工单系统相关安装介质如下：

组件名称	文件名	说明
运维工单系统	amp-workorder-v3.3.tar.gz	运维工单系统，SpringBoot应用

表6-1 运维工单系统产品介质 注意：安装后运行运维工单系统需要提供license.xml文件，请向您的产品服务提供商获取。

68 7.2 安装工单系统

69 7.2.1 安装准备

参考4.2.1节Web控制台的安装准备说明，如果已经安装相关依赖环境组件则可跳过本步骤。

70 7.2.2 安装说明

创建AMP产品安装根目录，指定${PATH} 为实际路径，将amp-workorder-v3.3.tar解压到对应目录及完成产品包安装，/${PATH}/AMP/amp-workorder为产品解压后的目录。

```
tar -zxvf amp-workorder-v3.3.tar.gz -C /${PATH}/AMP
```

如上，即完成运维工单系统的解压工作，接下来修改相关参数配置。

71 7.2.3 配置参数

修改amp-workorder/conf/application.yml文件，该配置文件为SpringBoot应用的默认配置文件，active的值为prod，其对应生效的文件是application-prod.yml,采用的是MySQL数据库连接配置。 系统中提供如下可选的配置文件:

文件名	说明
application-dev.yml	H2数据库作为持数据持久化存储的配置，开发环境阶段使用，生产环境不建议使用
application-prod.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-samle-dm.yml	达梦数据库作为数据持久化存储的配置
application-sample-gbase8s.yml	南大通用Gbase8s作为数据持久化存储的配置
application-sample-kingbasees.yml	人大进仓KingbaseES作为数据持久化存储的配置
application-sample-shentong.yml	神舟通用数据库作为数据持久化存储的配置

表6- 2运维工单系统配置文件

用户可根据实际部署环境修改application.yml文件中的active值为prod、sample-kingbasees、sample-dm、sample-shentong、sample-gbase-8s来切换不同环境的配置。 下面以采用MySQL配置的application-prod.yml文件为例，说明相关主要参数配置。

 server.port 参数指定了该运维工单系统的默认端口，默认值为9014

 spring.redis 指定了应用连接Redis相关配置，需要根据实际部署环境进行修改。

- timeout 为超时时间，默认3600s
- host为redis的ip地址，默认值localhost
- port为redis端口，默认值6379
- password 为redis连接密码

 spring.datasource 为数据库连接配置， 需要根据实际部署环境进行修改。

- url为数据库JDBC连接配置，包含数据库地址、端口、数据库名称等参数
- Username 指定数据库连接用户名
- Password 指定数据库连接密码

 amp为相关连接配置信息，需要根据实际部署环境进行修改。

- amp.consol.url 为web控制台应用的部署地址，默认http://localhost:9000

配置参考样例如下:

```
    redis:
      timeout: 3600
      host: localhost
      port: 6379
      password: root
    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
     amp:
      console:
```

```
    url: http://localhost:9000
    <省略其他>
```

72 7.2.4 初始化数据库

登录mysql数据库。执行amp-workorder/sql/mysql目录下的create.sql数据库创建脚本文件，initial.sql数据库初始化脚本文件，初始化运维工单系统。

初始化完成后输入exit退出mysql数据库。

```
create database amp_workorder;
use amp_cmdb;
source /${PATH}/AMP/ amp-workorder /sql/mysql/create.sql;
source /${PATH}/AMP/ amp-workorder/sql/mysql/initial.sql;
```

73 7.3 安装后的工作

74 7.3.3 了解产品目录结构

目录	包含
bin	工单管理平台组件的启动脚本。
conf	一些配置文件。
lib	应用程序依赖的一些jar包。
sql	监控平台对应的amp_workorder数据库多种版本的sql创建及初始化脚本文件。
HELP.md	帮助文档，对工单管理平台项目的补充说明。

表 6 3工单管理平台目录结构

75 7.3.2 启动运行服务

1. 修改完amp-workorder的配置文件后，后台启动Web控制台。

   nohup /${PATH}/AMP/amp-workorder/bin/startup.sh &

2. 查看Web控制台运行状态，若端口9000存在，表示启动成功。

   netstat –lntp | grep 9014

76 7.3.3 注册为平台服务

运维工单系统部署完成并注册为平台服务后，用户才可以访问使用。 登录访问Web控制台，选择【平台管理】->【服务管理】。编辑【运维工单系统】服务URL址。修改完成，点击服务上线，完成运维工单系统的注册和上线。

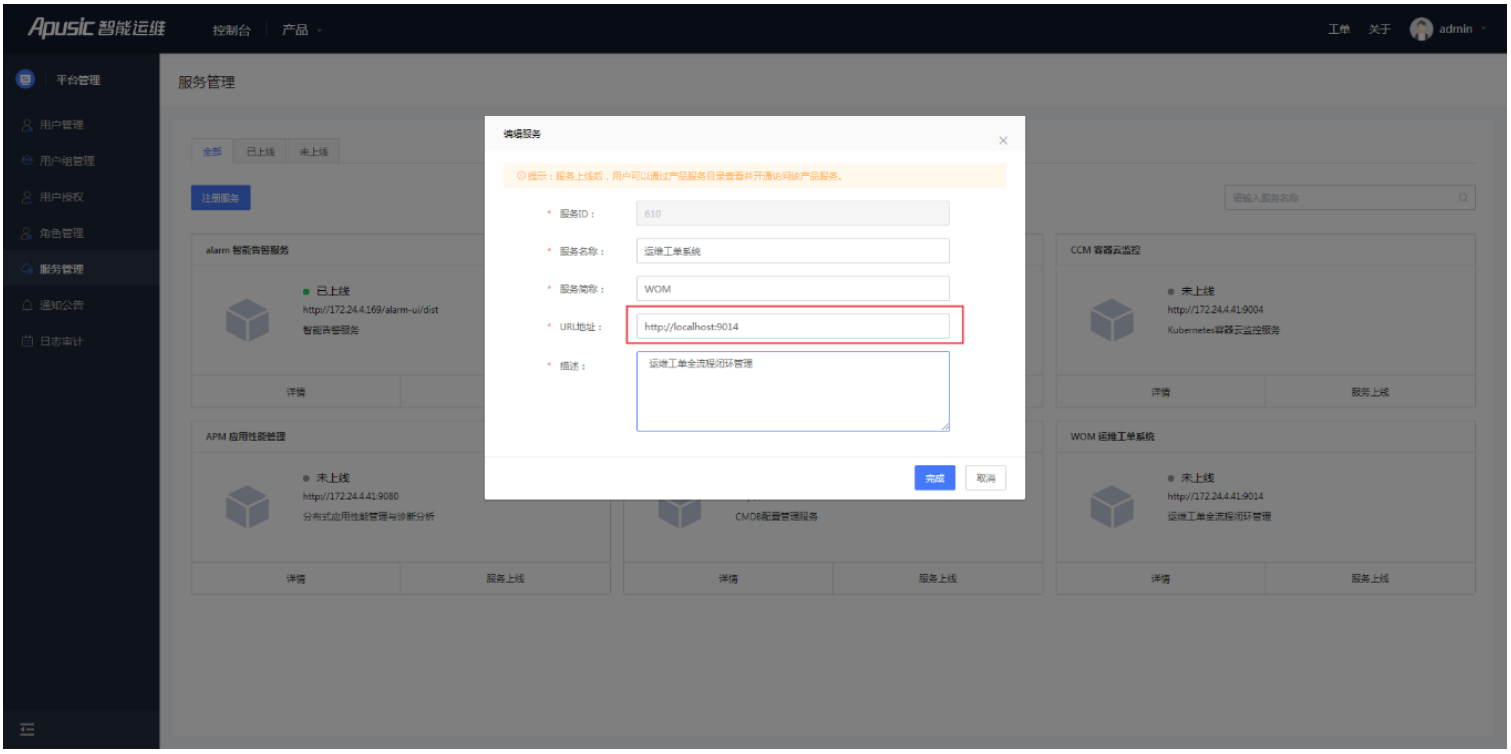


图6-1注册工单服务 切换至运维工单系统服务，出现如下图则表明部署成功 。

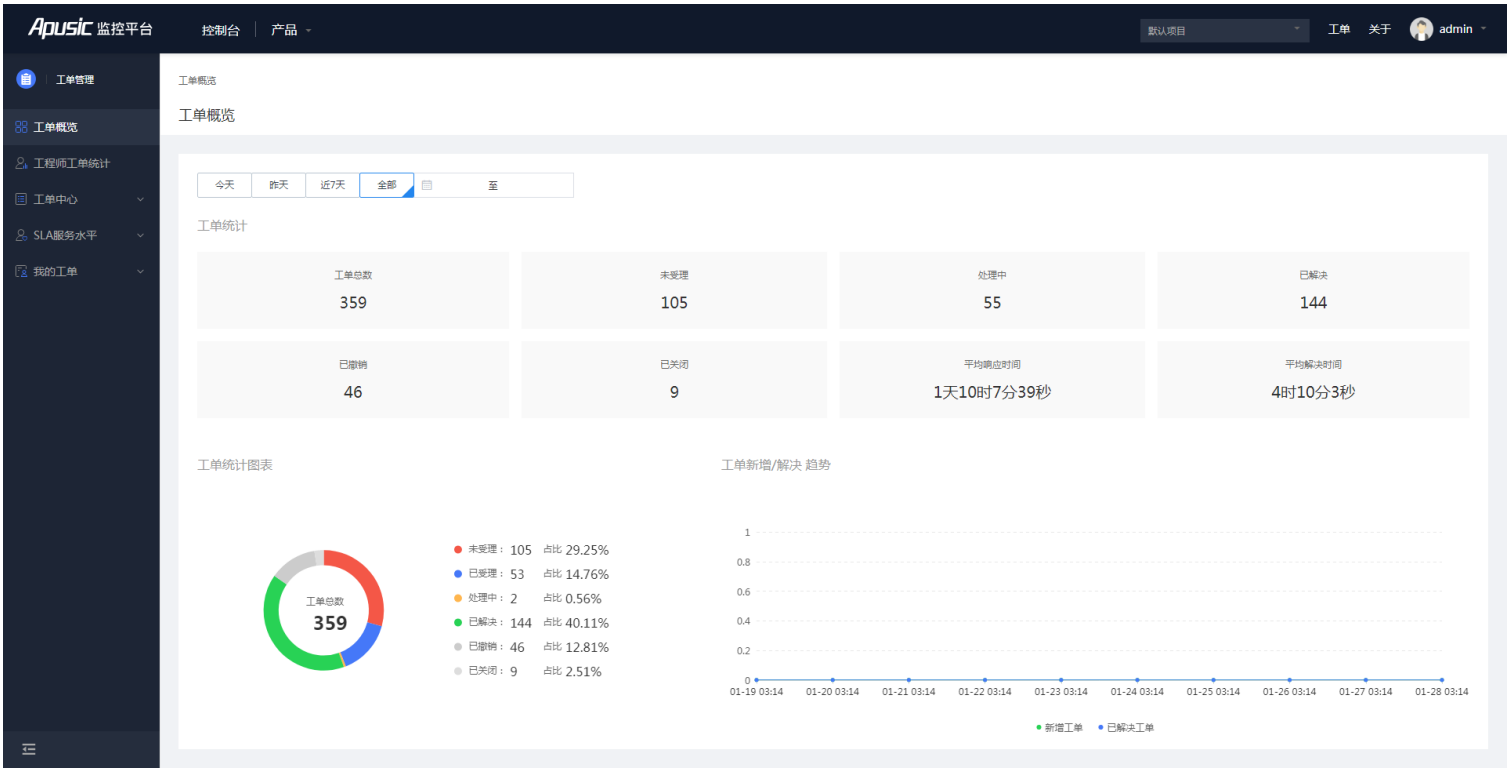


图 6-2 运维工单系统首页

77 7.3.4 停止运行服务

查看端口9014所运行的进程的pid，使用kill命令终止运维工单系统进程。

```
# netstat -lntp | grep 9014
tcp6       0      0 :::9014       :::*            LISTEN      19358/java
# kill -9 19358
```

78 7.3.5 卸载服务

执行下列命令，删除运维工单系统安装部署目录，即可完成卸载操作。

```
rm -rf /${PATH}/AMP/amp-workorder*
```

79 第8章 安装容器云监控服务

容器云监控服务实现对Kubernetes集群及workload、pod的观测，支持多集群统一纳入监控观测。容器云监控服务是AMP平台可选的产品服务，如果用户不需要该服务，可以跳过本章节。

80 8.1 产品介质说明

1. 容器云监控服务产品相关安装介质文件： □ amp-cloud-monitor-v3.3.tar.gz □ amp-monitor-kubernetes.linux-amd64.tar.gz
2. 将amp-monitor-kubernetes.linux-amd64.tar.gz产品安装包解压后，可以看到下面几个具体的安装包文件： 文件名 说明 node-exporter.tar 服务器基本指标收集的组件 kube-state-metrics.tar 监控k8s资源对象的组件 amp-monitor-agent.tar 监控服务器组件，包括Prometheus监控服务引擎和monitor-agent监控代理程序 node-exporter.yml node-exporter.tar的启动配置文件 kube-state-metrics.yml kube-state-metrics.tar的启动配置文件 amp-monitor-agent.yml amp-monitor-agent.tar的启动配置文件 表 7-1 容器云监控K8S插件文件列表 注意：安装后运行容器云监控服务需要提供license.xml文件，请向您的产品服务提供商获取。

81 8.2 安装容器云监控服务

82 8.2.1 安装准备

参考4.2.1节Web控制台的安装准备说明，如果已经安装相关依赖环境组件则可跳过本步骤。

83 8.2.2 安装说明

将amp-cloud-monitor-v3.3.tar.gz解压到对应目录及完成产品包安装，/${PATH}/AMP/ amp-cloud-monitor为产品解压后的目录。

84 8.2.3 配置参数

1. 修改amp-cloud-monitor/conf/application.yml文件，该配置文件为SpringBoot应用的默认配置文件，active的值为prod，其对应生效的文件是application-prod.yml,采用的是MySQL数据库连接配置。 系统中提供如下可选的配置文件:

文件名	说明
application-dev.yml	H2数据库作为持数据持久化存储的配置，开发环境阶段使用，生产环境不建议使用
application-prod.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-samle-dm.yml	达梦数据库作为数据持久化存储的配置
application-sample-gbase8s.yml	南大通用Gbase8s作为数据持久化存储的配置
application-sample-kingbasees.yml	人大进仓KingbaseES作为数据持久化存储的配置
application-sample-shentong.yml	神舟通用数据库作为数据持久化存储的配置

表8-2 容器云监控应用配置文件

2. 用户可根据实际部署环境修改application.yml文件中的active值为prod、sample-kingbasees、sample-dm、sample-shentong、sample-gbase-8s来切换不同环境的配置。 下面以采用MySQL配置的application-prod.yml文件为例，说明相关主要参数配置。

 server.port 参数指定了该容器云监控的默认端口，默认值为9004

 spring.redis 指定了应用连接Redis相关配置，需要根据实际部署环境进行修改。

- timeout 为超时时间，默认3600s
- host为redis的ip地址，默认值localhost
- port为redis端口，默认值6379
- password 为redis连接密码 spring.datasource 为数据库连接配置， 需要根据实际部署环境进行修改。
  - url为数据库JDBC连接配置，包含数据库地址、端口、数据库名称等参数 - - Username 指定数据库连接用户名 - - Password 指定数据库连接密码

 amp为相关连接配置信息，需要根据实际部署环境进行修改。

- amp.consol.url 为web控制台应用的部署地址，默认http://localhost:9000 配置参考样例如下:

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

   datasource:
     type: com.zaxxer.hikari.HikariDataSource
     url: jdbc:mysql://localhost:3306/amp-cloud-monitor
   ?
useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=U
     username: root
     password: root
   amp:
     console:
```

```
    url: http://localhost:9000
  <省略其他>
```

85 8.2.4 初始化数据库

1. 登录mysql数据库。执行amp-cloud-monitor/sql/mysql目录下的create.sql数据库创建脚本文件，initial.sql数据库初始化脚本文件，初始化web容器云监控数据库。

```
create database cloud_monitoring;
use cloud_monitoring;
source /${PATH}/AMP/amp-cloud-monitor/sql/mysql/create.sql;
source /${PATH}/AMP/amp-cloud-monitor/sql/mysql/initial.sql;
```

2. 连接上amp_console数据库，修改容器云监控服务访问地址，即部署服务应用的ip及端口号，修改服务url为实际部署的地址。

   use amp_console； update `amp_console`.`srv` set url = 'http://localhost:9004' where id =2;

3. 初始化完成后输入exit退出mysql数据库。

86 8.3 安装Kubernetes监控插件

87 8.3.3 注册kubernets集群

1. 前置工作，用户已经搭建了Kubernetes集群。
2. 使外部网络可以访问k8s的内部资源对象，设置代理，使外部网络可以通过8888端口访问k8s资源对象。

   nohup kubectl proxy --address='0.0.0.0' --accept-hosts='^*$' --port=8888 &
3. 登录AMP监控平台，注Kubernetes集群，如下图7-3所示。

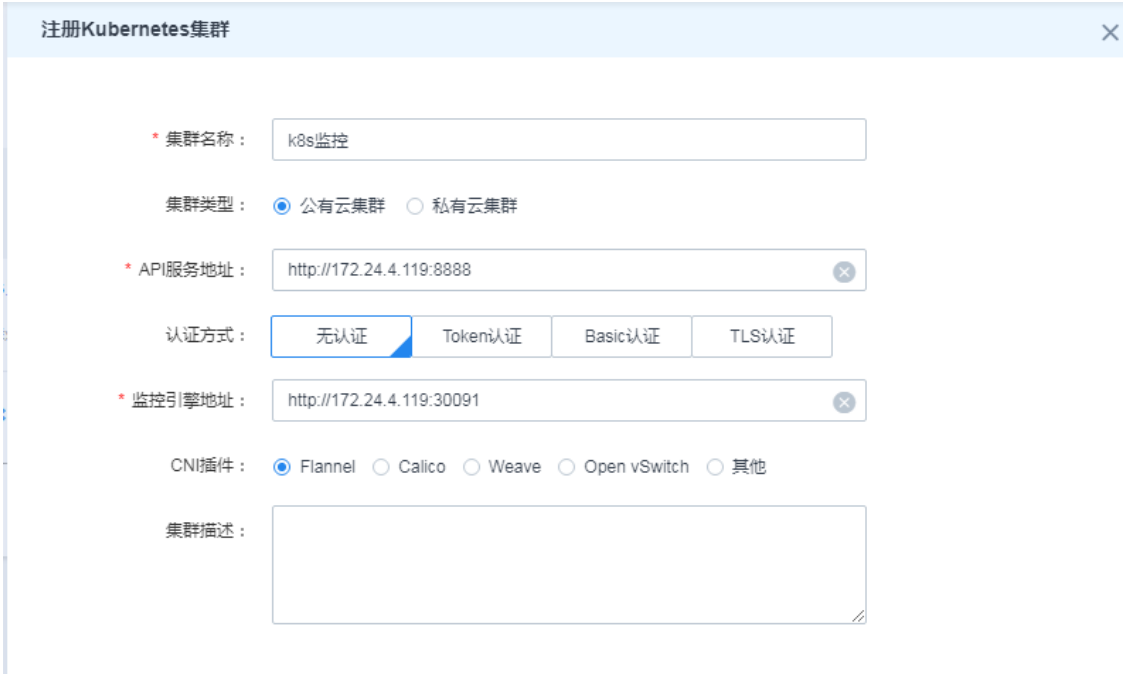


图 7-1 注册kubernetes集群界面

1. 监控引擎：选择之前添加的容器类型监控引擎。
2. API服务地址：添加上一步暴露给外部网络访问k8s集群资源的访问地址。
3. 其他参数根据实际情况填写，记录注册的kubernetes集群的ID, 后续操作会使用到。

88 8.3.2 安装监控插件前准备工作

1. 创建k8s目录,存放k8s安装介质包。

   mkdir –p /${PATH}/K8S

2. amp-monitor-kubernetes.linux-amd64.tar.gz安装介质包并解压到K8S目录：

   tar –zxvf amp-monitor-kubernetes.linux-amd64.tar.gz –C /path/K8S

3. 解压后得到node-exporter.tar、kube-state-metrics.tar、amp-monitor-agent.tar这3个产品包和node-exporter.yml、kube-state-metrics.yml、amp-monitor-agent.yml这3个yml启动文件。

4. 如果用户自己有已经搭建的镜像仓库，可以将上述3个镜像文件上传至镜像仓库，之后集群中的每个节点加载镜像文件进行安装操作，否则，用户需要手动将3个镜像文件拷贝至集群中的每个节点。

5. 在集群中一个节点安装举例如下，解压产品包安装后，加载相应的镜像文件。

```
#加载镜像文件
#加载node-exporter.tar
docker load –i node-exporter.tar
#加载kube-state-metrics.tar
docker load –i kube-state-metrics.tar
#加载amp-monitor-agent.tar
docker load –i amp-monitor-agent.tar
```

89 8.3.3 安装node-exporter

1. 获取产品包解压后的node-exporter的配置文件node-exporter.yml。

   $ kubectl apply –f node-exporter.yml

2. 执行以下命令安装node-exporter

90 8.3.4 安装kube-state-metrics

1. 获取产品包解压后的kube-state-metrics.tar的配置文件kube-state-metrics.yml。

2. 执行以下命令安装kube-state-metrics。

91 8.3.5 安装monitor-agent

用户如果想实现监控k8s集群资源，若达到相应的报警策略条件，触发告警信息，则需要部署amp-monitor-agent服务。

92 8.3.5.1 修改monitor-agent启动配置文件

1. 获取产品包解压后的amp-monitor-agent-amd64.tar的配置文件amp-monitor-agent-amd64.yml，修改相应的内容，实现监控告警的功能。

```
vi /path/K8S/amp-monitor-agent-amd64.yml
```

2. 修改data.prometheus.yml配置内容;

1. 将amp_k8s_id修改为用户在AMP平台注册的k8s集群的id。
2. 将targets修改为prometheus实际的访问地址。
3. 将所有的amp_k8s_id对应的replacement的值全部替换为对应的amp_k8s_id，有多处需要修改。

3. 修改kws_agent_standalone.conf配置内容;

1. 修改console_url为实际控制台组件的访问地址。
2. 修改kws_app_url为实际监控平台组件的访问地址。
3. 修改prom_alertmanager_url为实际AALARM的告警通知地址（ip+端口）。

```
    apiVersion: v1
    data:
      kws_agent_standalone.conf: |-

    basic_password = kube@0514
    #
    basic_user_name = kubetest
    #blackbox地址
    blackbox_export_url = kws-blackbox-9115-300037.kcssz.cloud.kingdee.com
    #控制台url
    #console_url = http://cloud.kingdee.com
    console_url = http://172.24.4.115:9000

    #监控服务app地址
    kws_app_url = http://172.24.4.115:9002
    #
    load_sign = smallxiu

    #是否输出到控制台
    log_console = 1

    #日志打印
    log_level = 1
    #alertmanager地址
prom_alertmanager_url = 172.24.4.115:9093

    #远程存储读url
    #remote_read_url = http://172.24.2.62:7201/api/v1/prom/remote/read
```

```
    #远程存储写url
    #remote_write_url = http://172.24.2.62:7201/api/v1/prom/remote/write

kind: ConfigMap
metadata:
  name:  kws-agent-standalone-config
```

4. 修改app.conf配置内容;

    1. 修改prome_id为实际部署的监控k8s集群的那个prometheus在监控平台注册后的id。

    apiVersion: v1 data: app.conf: |- AppName=kws-agent #运行模式 runmode=agent_standalone #httpport端口 httpport=8100 #区域id #region_id=4 #普罗米修斯id prome_id=3001 #日志打印 log_level=1 #是否输出到控制台 log_console=0 prome_reload_url=http://localhost:9090/-/reload

5. 修改端口访问配置内容;

1. 修改Spec.ports.name:prometheus.nodePort端口为其他用户自定义的端口，默认30090，可以不进行修改。

2. 修改Spec.ports.name:agent.nodePort端口为其他用户自定义的端口，默认30091，可以不进行修改。

```
apiVersion: v1
kind: "Service"
metadata:
  name: monitor-agent
  labels:

    name: monitor-agent
spec:
  ports:
  - name: prometheus
    protocol: TCP
    port: 9090
    targetPort: 9090
    nodePort: 30090
  - name: agent
    protocol: TCP
    port: 8100
    targetPort: 8100
    nodePort: 30091
  selector:
    app: monitor-agent
```

93 8.3.5.2 运行monitor-agent

1. 进入k8s集群环境中，将上一步添加的配置文件上传到k8s集群服务器上，执行下列操作，运行agent服务。

```
kubectl apply -f amp-monitor-amd64.yml
```

94 8.4 安装后的工作

95 8.4.1 了解产品目录结构

目录	包含
bin	监控平台组件的启动脚本。
conf	一些配置文件。
lib	应用程序依赖的一些jar包。
sql	监控平台对应的cloud_monitoring数据库多种版本的sql创建及初始化脚本文件。
HELP.md	帮助文档，对容器云监控平台项目的补充说明。

表 7 3 amp-cloud-monitor监控平台目录结构

96 8.4.2 启动运行服务

修改完amp-cloud-monitor的配置文件后，后台启动Web控制台。

```
nohup /${PATH}/AMP/amp-cloud-monitor/bin/startup.sh &
```

查看Web控制台运行状态，若端口9004存在，表示启动成功。

```
netstat -lntp | grep 9004
```

97 8.4.3 注册为平台服务

容器云监控服务部署完成并注册为平台服务后，用户才可以访问使用 登录访问Web控制台，选择【平台管理】->【服务管理】。编辑【容器云监控】服务URL址。修改完成，点击服务上线，完成容器云监控服务的注册和上线

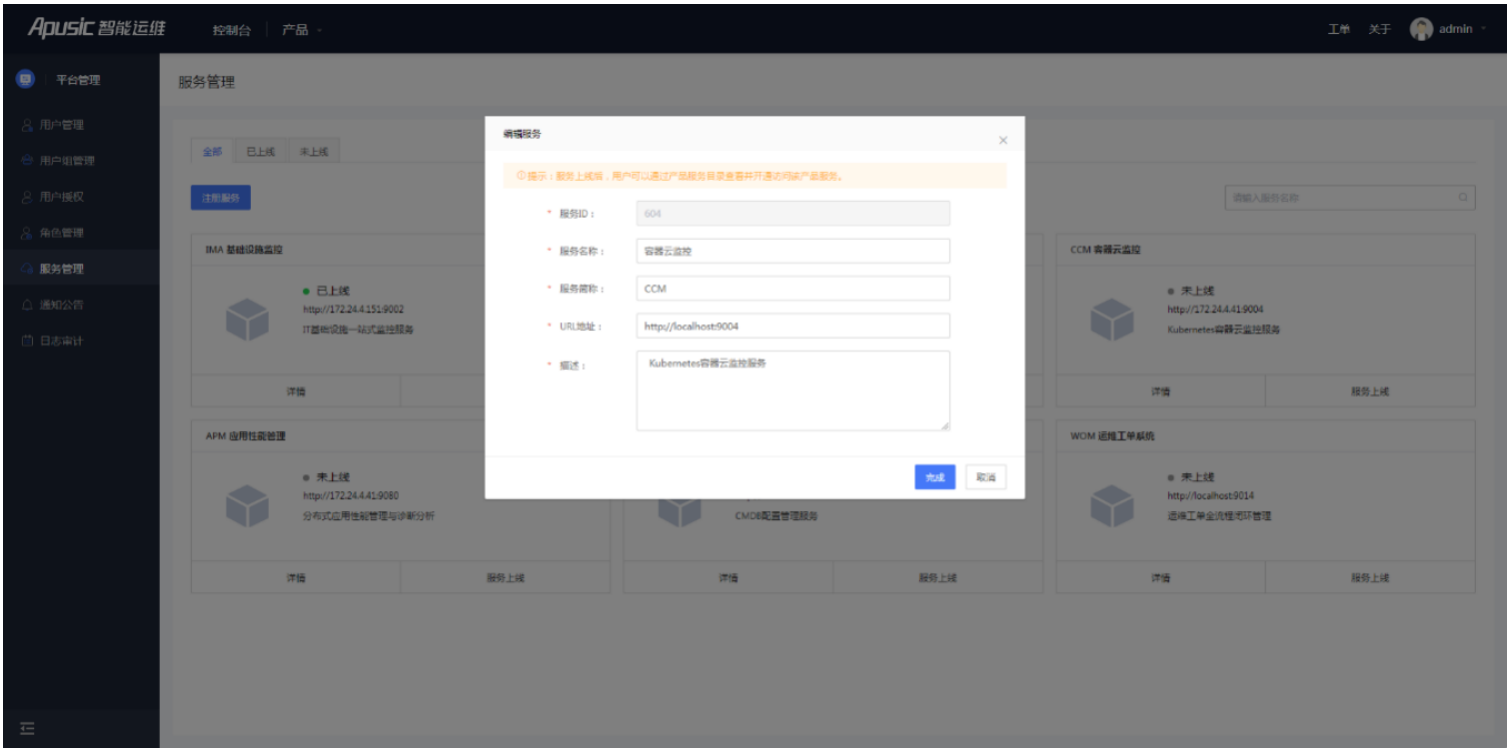


图7-2 注册容器云监控服务 切换至容器云监控服务，出现如下图7-3则表明部署成功 。

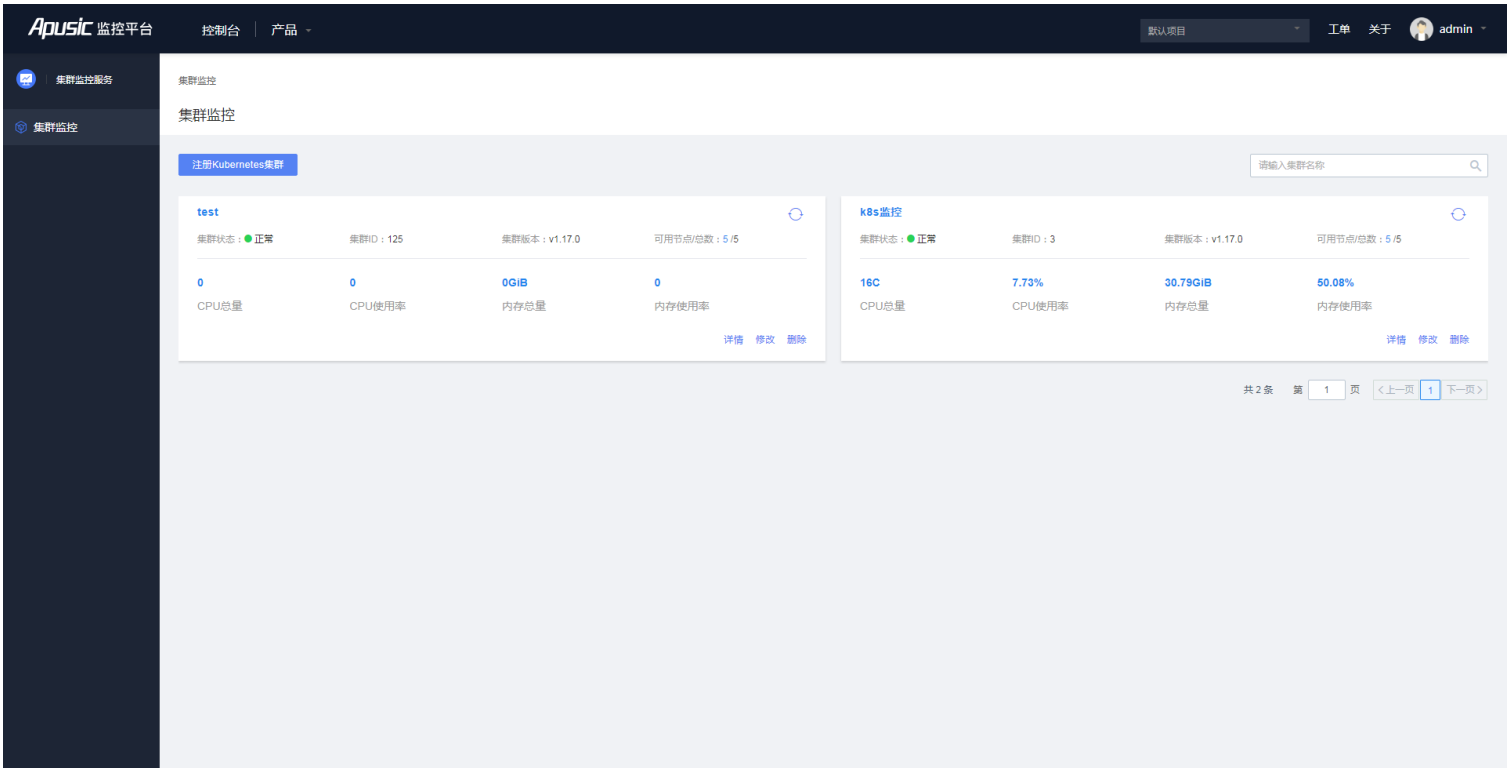


图7-3 容器云监控服务首页

98 8.4.4 停止运行服务

查看端口9004所运行的进程的pid，使用kill命令终止容器云监控服务进程。

```
# netstat -lntp | grep 9004
tcp6       0      0 :::9004       :::*            LISTEN      19358/java
# kill -9 19358
```

99 8.4.5 卸载服务

执行下列命令，删除容器云监控安装部署目录，即可完成卸载操作。

```
rm -rf /${PATH}/AMP/amp-cloud-monitor
```

100 第9章 附录：集群高可用安装

101 9.1 集群高可用部署架构

AMP产品集群高可用部署，各组件调用关系如下图所示。

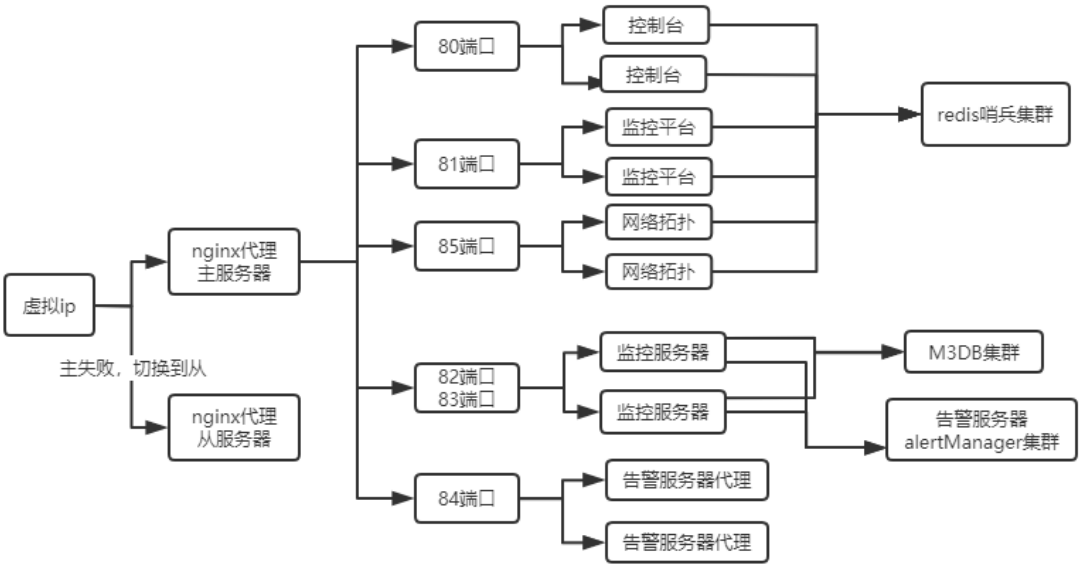


图9-1 AMPV3.3 各组件高可用部署调用关系图

- nginx高可用：Nginx结合keepalived实现nginx高可用，若一个为master的nginx服务器出现宕机，keepalived会将所有的请求转移到从的nginx服务器上。
- redis高可用：通过redis哨兵模式，控制台缓存数据，监控平台缓存数据，网络拓扑网络设备关系数据，通过redis哨兵集群保存。
- 告警服务器高可用：通过部署alertManager集群，集群内部通过Gossip实现高可用，告警只会成功发送一次。
- AMP组件高可用：控制台，监控平台，监控引擎，监控服务器，告警服务器代理，网络拓扑等组件的高可用通过nginx的端口代理实现。
- M3DB集群存储：时序数据使用M3DB集群进行存储，实现监控数据的高可用。
- 数据库唯一：多个控制台组件使用同一个console数据库，多个监控平台使用同一个monitoring数据库。

```
AMP V3.3中所用组件实现高可用在下面进行简单说明。

    组件 高可用方案          备注说明
    Nginx       Nginx+keepalived双机主从模式       通过虚拟ip访问。
    Redis       Redis 哨兵        一主，二从，三哨兵
    AMP控制台（console）  Nginx反向代理     代理端口：80
    AMP监控平台(monitor)  Nginx反向代理     代理端口：81
    监控服务器（agent）    Nginx反向代理     包括：agent，监控引擎
    分别通过82端口，83端口代理
    告警服务器（alarm）    Nginx反向代理     代理端口：84
    网络拓扑（nettopo）    Nginx反向代理     代理端口：85
    M3DB集群      M3DB集群方式       通过M3DB集群实现监控数据的高可用
```

102 9.2 Nginx高可用部署

1. Nginx安装过程见附录，Nginx的版本为1.16.1。Keepalived安装过程如下

```
yum install -y keepalived
```

2. 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。

3. 新建检查nginx时候运行的脚本文件。

```
vi /root/check_nginx.sh
```

4. 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。

```
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
```

5. 修改检查文本为可执行文件。

   chmod +x /root/check_nginx.sh

6. 编辑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。

   vi /etc/keepalived/keepalived.conf

7. 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
 }
```

8. 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
   }
}
```

9. 关闭防火墙，从主节点启动keepalived

   service keepalived start

10. 通过访问nginx的80端口，可以实现访问。

11. 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。

12. 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。

103 9.3 Redis高可用部署

通过部署Redis的哨兵模式集群，实现高可用。推荐版本，Redis的版本是5.0及以上。（这里使用5.0.4版本演示）

1. 搭建步骤，准备三台服务器： 一个主服务器、二个从服务器、三个哨兵：

```
172.24.4.110
172.24.4.111
172.24.4.112


服务类型          是否主服务器       ip地址    端口
Redis  是       172.24.4.110    6379
Redis  否       172.24.4.111    6379
Redis  否       172.24.4.112    6379
Sentinel(哨兵)          172.24.4.110    26379
Sentinel                172.24.4.111    26379
Sentinel                172.24.4.112    26379
```

2. 我们统一安装在在3台服务器/usr/local/redis目录下,安装redis。

```
    yum -y install gcc
    yum -y install gcc-c++
    wget http://download.redis.io/releases/redis-5.0.4.tar.gz
    tar -zxvf redis-5.0.4.tar.gz
    mv redis-5.0.4  /usr/local/redis
    cd /usr/local/redis
    make
    make install
```

3. 编辑主服务器172.24.4.110的redis.conf配置文件。

```
    vim /usr/local/redis/redis.conf
```

4. 修改如下内容。

```
    port 6379
    # 使得Redis可以跨网络访问
    bind 0.0.0.0
    # 设置为后台启动
    daemonize yes
    # 禁止保护模式
    protected-mode no
```

5. 编辑两台从服务器172.24.4.111,172.24.4.112的redis.conf配置文件。

```
    vim /usr/local/redis/redis.conf
```

配置两份从服务器172.24.4.111,172.24.4.112的redis.conf配置文件，修改内容如下：

```
# 服务器端口号
port 6379
# 使得Redis可以跨网络访问
bind 0.0.0.0
# 设置为后台启动
daemonize yes
# 禁止保护模式
protected-mode no
# 下面配置只需要配置从节点(slave)
# 配置主服务器地址、端口号
replicaof 172.24.4.110 6379
```

分别启动这三台服务器的Redis服务。

```
/usr/local/redis/src/redis-server /usr/local/redis/redis.conf
```

将sentinel.conf文件复制两份并分别修改配置，编辑配置文件/usr/local/redis/sentinel.conf。

```
vi /usr/local/redis/sentinel.conf
```

修改内容如下：

```
# 三个配置文件填写相应的端口号，目前均设置为26379
port 26379
# 设置为后台启动
daemonize yes
# 禁止保护模式
protected-mode no
# 使得Redis可以跨网络访问
bind 0.0.0.0
# 主节点的名称(可以自定义，与后面的配置保持一致即可)
# 主机地址
# 端口号
# 数量(2代表只有两个或两个以上的哨兵认为主服务器不可用的时候，才会进行failover操作)
#sentinel monitor mymaster 127.0.0.1 6379 2
sentinel monitor mymaster 172.24.4.110 6379 2

# 多长时间没有响应认为主观下线(SDOWN)
sentinel down-after-milliseconds mymaster 30000
# 表示如果15秒后,mysater仍没活过来，则启动failover，从剩下从节点序曲新的主节点
sentinel failover-timeout mymaster 15000
# 指定了在执行故障转移时， 最多可以有多少个从服务器同时对新的主服务器进行同步， 这个数字越小， 完成故障转移所需的时间就越长
sentinel parallel-syncs mymaster 1
```

分别启动三台服务器上的sentinel。

```
/usr/local/redis/src/redis-sentinel /usr/local/redis/sentinel.conf
```

登录172.24.4.110:6379查看哨兵集群信息 登入redis-cli客户端后输入info replication查看集群关系。

```
/usr/local/redis/src/redis-cli -h 172.24.4.110 -p 6379
```

```
info replication #查看该节点的角色信息
```

结果信息如下。

```
# Replication
role:master        #该节点的角色 master-主节点， slave-从节点
connected_slaves:2     #从节点的个数
slave0:ip=172.24.4.111,port=6379,state=online,offset=6026671,lag=1
slave1:ip=172.24.4.112,port=6379,state=online,offset=6026671,lag=1
master_replid:18df788d90dc022b74a609be236fff0833b45868
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:6026671
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:5952111
repl_backlog_histlen:74561
```

然后手动关闭主节点的redis服务后，查看两个slave节点信息，发现其中一个变成了master。

104 9.4 AMP组件高可用部署

AMP各组件高可用通过部署多个组件实例，通过访问Nginx的反向代理和负载均衡实现各组件的高可用，AALARM则是通过部署集群实现高可用。

105 9.4.1 修改Nginx的配置文件

1. 修改在目录下的/usr/local/nginx/conf/nginx.conf配置文件，在http块中加入以下内容：下面console, monitor, Prometheus, agent, alarm, nettop分别对应的是80， 81， 82, 83， 84， 85,端口，这个是默认设置的，可以进行修改，若进行修改，后面组件的配置文件也需要进行相应的修改，不推荐修改。在下面配置文件中，默认每个组件的服务是两个，可以进行组件数量的增加或修改，将下面各组件的ip1，ip2替换为实际组件部署的ip地址。

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

#监控平台
    upstream monitor {
        server monitor_ip1:9002;
        server monitor_ip2:9002;
    }

    #agent服务 监控引擎代理
    upstream prometheus{
        server agent_ip1:9090;
        server agent_ip2:9090;
    }

    #agent服务
    upstream agent{
        server agent_ip1:8100;
        server agent_ip2:8100;
    }
```

#网络拓扑 upstream nettopo { server nettopo_ip1:9808; server nettopo_ip2:9808; } server { listen 80; server_name localhost; location / { proxy_pass http://console; } } server { listen 81; server_name localhost; location / { proxy_pass http://monitor; } } server { listen 82; server_name localhost; location / { proxy_pass http://prometheus; } }

```
server {
    listen       83;
    server_name  localhost;
    location / {
        proxy_pass http://agent;
    }
}

server {
    listen       84;
    server_name  localhost;
    location / {
        proxy_pass http://alarm;
```

```
    }
}

server {
    listen       85;
    server_name  localhost;
    location / {
        proxy_pass http://nettopo;
    }
}
```

106 9.4.2 控制台组件高可用

1. 先安装好数据库，搭建好Redis的集群。这里AMP的安装目录统一在/usr/local/AMP目录下。
2. 下面以mysql配置文件为例，其他数据库类似。参考第4章中安装控制台组件的步骤。
3. 编辑amp-console/conf/application-prod.yml配置文件

   vi /usr/local/AMP/amp-console/conf/application-prod.yml
4. 修改mysql数据库连接信息，将单节点部署中配置文件中的redis部署修改为以下方式，cluster.nodes的值分别是redis集群的3个，节点的访问地址（ip+哨兵端口），根据实际情况进行配置。master的值为哨兵集群的主服务器名称，和redis哨兵配置文件保持一致，不用进行修改。

   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
5. 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

   vi /usr/local/nginx/conf/nginx.conf
6. 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;
            location / {
                proxy_pass http://console;
            }
    }
```

7. 至此，控制台组件高可用部署完毕。

107 9.4.3 基础设施监控服务高可用

108 9.4.3.3 部署M3DB集群环境

监控服务器组件进行高可用时，多个监控引擎使用同一个数据源，AMP平台使用M3DB作为远程存储，多个监控引擎从M3DB集群中读取数据，M3DB使用集群的方式进行部署，具体的部署参考附录，部署成功后，需要使用M3DB集群存储数据。部署多个agent监控服务器，多个监控服务器使用相同的远程存储。

109 9.4.3.2 添加监控引擎代理访问地址

1. 添加监控引擎代理地址，在成功安装monitoring监控平台，登录监控平台后，在监控引擎管理界面，添加监控引擎代理地址时，填写监控引擎代理的访问路径：
2. 添加成功后，记录该条记录的监控引擎代理地址的id。 8.4.3.3 Web监控平台组件高可用
3. 参考第5章中安装监控平台组件的步骤，修改amp-monitoring/conf/application-prod.yml配置文件

   #http://nginx_proxy_ip:82 http://172.24.5.137:82
4. 修改如下几项信息：

1. 修改redis的集群连接节点。

   vi /usr/local/AMP/amp-monitoring/conf/application-prod.yml
2. 修改console的url为nginx的ip+80访问端口。
3. 修改agent的url为nginx的ip+83访问端口。（后续会进行部署）
4. 修改alarm的url为nginx的ip+84访问端口。（后续会进行部署）
5. 修改net_topo的url为nginx的ip+85访问端口。（后续会进行部署）
6. 修改mysql连接信息。
7. 将单节点部署中配置文件中的redis部署修改为以下方式，cluster.nodes的值分别是redis集群的3个
8. 节点的访问地址（ip+哨兵端口），根据实际情况进行配置。master的值为哨兵集群的主服务器名称，
9. 和redis哨兵配置文件保持一致，不用进行修改。

```
  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_monitoring
  ?
useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=U
    username: root
    password: root
  amp:
    console:
      url: http://localhost:80
    agent:
      url: http://localhost:83
  alarm:
      url: http://localhost:84
```

```
    net_topo:
      url: http://localhost:85
```

3. 监控平台组件可以部署多个，但是连接的数据库需要是同一个monitoring数据库。向nginx的配置文件中加入监控平台的配置，下面的配置是部署了两个监控平台组件，用nginx的81端口进行代理访问。

4. 至此，监控平台组件高可用部署完毕。

110 9.4.3.4 监控服务器高可用

1. 修改monitor-agent配置文件，参考第4章中安装监控服务器组件的步骤。

2. 编辑monitor-agent/runtime/config/kws_agent_standalone.conf配置文件

```
vi  /usr/local/AMP/monitor-agent/runtime/config/kws_agent_standalone.conf
```

3. 修改内容如下

1. 修改以下控制台地址，监控服务地址。

2. 修改alertManager访问的地址（alertManager后续会进行部署）。

3. 若alertManager未采用高可用部署，填写alertManager单节点地址。

```
#监控平台
upstream monitor {
    server monitor_ip1:9002;
    server monitor_ip2:9002;
 }
server {
           listen       81;
           server_name  localhost;
            location / {
                     proxy_pass http://monitor;
            }
    }
```

4. 若alertManager采用高可用部署，填写多个alertManager地址，用逗号分隔。

5. 使用M3DB进行存储，修改远程存储读url，远程存储写url。

```
#控制台url
#console_url = http://nginx_proxy_ip:80
console_url = http://172.24.5.137:80

#监控服务app地址
#kws_app_url = http://nginx_proxy_ip:81
kws_app_url = http://172.24.5.137:81

#alertmanager地址
#prom_alertmanager_url = 172.24.4.111:9093, 172.24.4.112:9093, 172.24.4.113:9093
prom_alertmanager_url = 172.24.5.137:9093

#远程存储读url
#remote_read_url = http://172.24.2.62:7201/api/v1/prom/remote/read
```

```
#远程存储写url
#remote_write_url = http://172.24.2.62:7201/api/v1/prom/remote/write
```

4. 修改monitor-agent/src/agent/conf/app.conf配置文件

```
vi  /usr/local/AMP/monitor-agent/src/agent/conf/app.conf
```

5. 将配置文件中的prome_id修改为上一步添加保存的监控引擎代理的id，修改prometheus配置文件及报警规则文件等路径。

```
#添加上一步加入的监控引擎代理id
prome_id=3002
prome_reload_url=http://localhost:9090/-/reload

#prometheus配置文件路径  每台机器不一样  本地配置
prometheus_config_path=/usr/local/AMP/monitor-agent/prometheus/prometheus.yml
#报警配置文件路径
alert_config_path=/usr/local/AMP/monitor-agent/prometheus/alert.yml
#报警配置文件夹路径
alert_dir_path=/usr/local/AMP/monitor-agent/prometheus/alert
#存储历史配置文件夹路径
prometheus_bak_path=/usr/local/AMP/monitor-agent/prometheus/configbak
#存储证书目录
prometheus_cafile_path=/usr/local/AMP/monitor-agent/prometheu
```

111 9.4.3.5 网络拓扑高可用组件高可用

1. 在解压安装包后，修改安装包目录下的config.yaml配置文件。

   vi /usr/local/AMP/amp-nettopo-amd64/config.yaml

2. 修改如下内容

   1. 将enable修改为true,即采用redis进行存储。默认采用文件进行存储。

2. 若redis为单节点部署，nodes后填写单节点地址，

3. 若为哨兵集群方式部署，则填写多个地址，添加master主服务器名称。

```
redis:
  enable: false                                #修改为true
  nodes: 127.0.0.1:6379
#修改为redis集群ip+sentinel端口
#  nodes: 172.24.4.110:26379,172.24.4.111:26379,172.24.4.112:26379
#默认redis哨兵集群的主服务器名称，和redis哨兵配置文件保持一致，不用进行修改
#  master: mymaster
```

3. 部署多个网络拓扑组件，加入nginx的端口代理中。

```
vi /usr/local/nginx/conf/nginx.conf
```

4. 将网络拓扑组件的配置加入nginx的配置文件中。

```
#网络拓扑
    upstream nettopo {
        server nettopo_ip1:9808;
        server nettopo_ip2:9808;
        }
        server {
                listen       85;
                server_name  localhost;
                location / {
                        proxy_pass http://nettopo;
                }
        }
```

5. 至此，网络拓扑的高可用部署完毕。

112 9.4.3.6 添加nginx代理地址

1. 部署多个agent监控服务器，将agent监控服务器加入nginx的端口代理中。

```
vi /usr/local/nginx/conf/nginx.conf
```

2. 添加监控引擎代理地址，在nginx的配置文件中添加相应的内容。

```
#监控引擎代理
    upstream prometheus{
          server agent_ip1:9090;
          server agent_ip2:9090;
        }
    server {
                listen       82;
                server_name  localhost;
                location / {
                    proxy_pass http://prometheus;
                }
        }
```

3. 添加agent代理服务地址，在nginx的配置文件中添加相应的内容

```
#agent服务
    upstream agent{
          server agent_ip1:8100;
          server agent_ip2:8100;
        }
    server {
                listen       83;
               server_name  localhost;
                location / {
                        proxy_pass http://agent;
                }
        }
```

4. 至此，监控服务器agent的高可用部署完毕。

113 9.4.1 运维工单系统高可用

1. 参考第6章中安装监控平台组件的步骤，编辑amp-workorder/conf/application-prod.yml配置文件

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

2. 修改mysql数据库连接信息，将单节点部署中配置文件中的redis部署修改为以下方式，cluster.nodes的值分别是redis集群的3个，节点的访问地址（ip+哨兵端口），根据实际情况进行配置。master的值为哨兵集群的主服务器名称，和redis哨兵配置文件保持一致，不用进行修改。

```
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=U
  username: root
  password: root
amp:
  console:
    url: http://localhost:80
```

3. 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

```
vi /usr/local/nginx/conf/nginx.conf
```

4. 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 workorder {
    server workorder_ip_1:9014;
    server workorder_ip_2:9014;
 }

 server {
            listen       80;
            server_name  localhost;
            location / {
                proxy_pass http://workorder;
            }
    }
```

至此，控制台组件高可用部署完毕。

114 9.4.1 容器云监控高可用

1. 参考第7章中安装监控平台组件的步骤，编辑amp-cloud-monitor/conf/application-prod.yml配置文件

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

2. 修改MySQL数据库连接信息，将单节点部署中配置文件中的Redis部署修改为以下方式，cluster.nodes的值分别是Redis集群的3个，节点的访问地址（ip+哨兵端口），根据实际情况进行配置。master的值为哨兵集群的主服务器名称，和Redis哨兵配置文件保持一致，不用进行修改。

```
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=U
  username: root
  password: root
amp:
  console:
    url: http://localhost:80
```

3. 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

```
vi /usr/local/nginx/conf/nginx.conf
```

4. 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 amp-cloud-monitor {
    server amp-cloud-monitor_ip_1:9004;
    server amp-cloud-monitor_ip_2:9004;
 }

 server {
            listen       80;
            server_name  localhost;
            location / {
                proxy_pass http:// amp-cloud-monitor;
            }
    }
```

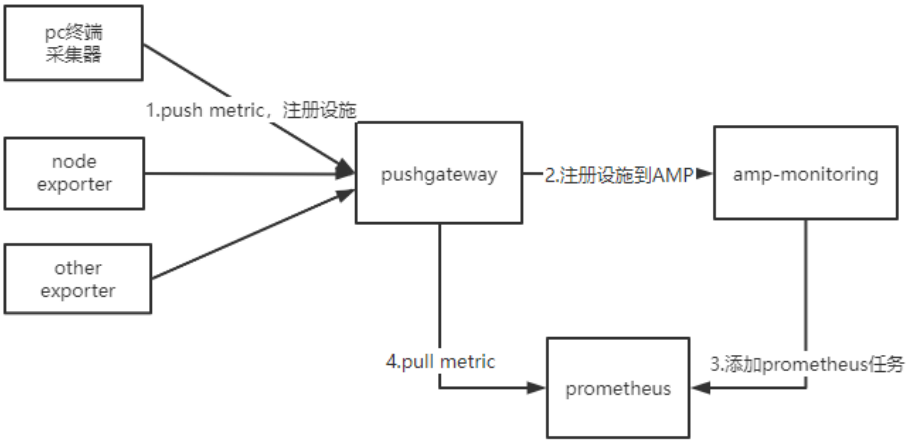
至此，控制台组件高可用部署完毕。

115 第10章 amp pushgateway安装使用

116 10.1 amp pushgateway介绍

amp pushgateway基于pushgateway开发，监控指标数据分别从pushgateway进行拉取，对于监控对象的过期指标数据进行过滤，不进行拉取。

下面是AMP pushgateway调用关系架构图



117 10.2 amp pushgateway使用

118 10.2.1 启动

下载并解压产品包

amp-pushgateway-amd64.tar.gz

pushgateway需要指定AMP基础监控地址，指定AMP数据中心，指定AMP监控引擎，因此需要修改下面参数。

修改start.sh文件

修改下面参数

```
PUSHGATEWAY_PORT=9091
AMP_MONITORING_URL="http://127.0.0.1:9002"
AMP_DATACENTER_ID=1
AMP_PROMETHEUS_ID=1
```

pushgateway组件端口PUSHGATEWAY_PORT一般使用默认不需要修改 修改AMP基础监控地址信息 AMP_MONITORING_URL="http://127.0.0.1:9002" 为具体的部署地址 修改AMP基础监控数据中心AMP_DATACENTER_ID参数的值为数据中心id 修改AMP基础监控数据中心AMP_PROMETHEUS_ID参数的值为监控引擎prometheus的id

执行启动脚本

```
./start.sh
```

119 10.2.2 关闭

执行停止采集器脚本

```
./stop.sh
```

120 10.2.3 附录 amp pushgateway 启动参数说明

pushgateway 启动参数详细说明

```
# 基础监控地址
amp.monitoring.url

# AMP 数据中心id
amp.monitoring.dataCenterId

# AMP 监控引擎id
amp.monitoring.prometheusId

# 部署pushgateway组件的服务器ip地址（一般默认不用修改，系统默认获取ip，如果获取存在问题，需要手动指定）
web.host-ip-address
```

查看更多的参数，可以使用./pushgateway --help命令进行查看。

121 第11章 附录：环境组件安装

122 11.1 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
```

123 11.2 安装MySQL

AMP的监控平台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。 是否覆盖？按y 回车

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

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。

124 11.3 安装Nginx

1. 到nginx官网http://nginx.org/en/download.html下载nginx-1.16.1.tar.gz安装包

2. 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
```

3. 进入放源码的目录下,将下载的nginx-1.16.1.tar.gz安装包放在该目录下

```
cd /usr/local/src
```

4. 解压nginx-1.16.1.tar.gz

```
tar -xvf nginx-1.16.1.tar.gz
```

5. 将解压包移动到/usr/local/nginx目录

```
mv nginx-1.16.1  /usr/local/nginx
```

6. 编译安装Nginx

```
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
```

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

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

8. 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
```

9. 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
```

125 11.4 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-5.0.4.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. 在 redis 目录下创建 bin 和 etc 两个文件夹。

```
mkdir -p /usr/local/redis/bin
mkdir -p /usr/local/redis/etc
```

8. redis-5.0.5 里的主配置文件 redis.conf 移动到刚创建的 etc 文件夹。

```
cd redis-5.0.5
mv redis.conf /usr/local/redis/etc/
```

9. 将 src 目录里带有绿色标识的文件全都移动到刚创建的 bin 文件夹。

```
cd src/
mv mkreleasehdr.sh redis-benchmark redis-check-aof redis-check-rdb redis-cli redis-sentinel
redis-server redis-trib.rb /usr/local/redis/bin/
```

10. 进入 etc 目录，修改 redis.conf 文件。

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

11. 注释掉 bind 127.0.0.1 这一行。

```
#bind 127.0.0.1
```

12. 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
```

13. 将 daemonize 属性改为 yes （这样启动时就在后台启动）。

```
daemonize yes
```

14. 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
```

15. 在 redis 目录下执行,启动redis，查看redis是否成功启动。

```
cd /usr/local/redis/
./bin/redis-server /usr/local/redis/etc/redis.conf
ps -ef | grep redis
```

126 11.5 安装M3DB单节点

127 11.5.1 安装前的准备

1. 关闭防火墙

```
systemctl status firewalld
systemctl stop firewalld
systemctl disable firewalld
systemctl status firewalld
```

2. 用vi编辑/etc/selinux/config文件，关闭SELinux，更改完成后保存退出。

```
SELINUX=disabled
```

3. 内核内置

```
sysctl -w vm.max_map_count=3000000
sysctl -w vm.swappiness=1
sysctl -w fs.file-max=3000000
sysctl -w fs.nr_open=3000000
```

4. 建立缓存目录

```
mkdir -p /var/lib/m3kv
```

5. 安装jq

```
yum install jq
```

128 11.5.2 存储节点安装

1. 二进制包安装

1. 在https://github.com/m3db/m3/releases网址下载m3_0.14.2_linux_amd64.tar.gz压缩包。

2. 解压tar -xvf m3_0.14.2_linux_amd64.tar.gz。

```
tar -xzvf m3_0.14.2_linux_amd64.tar.gz
```

3. 执行文件拷贝。

```
cp m3_0.14.2_linux_amd64/m3dbnode /usr/local/bin
```

2. 配置文件

1. 创建配置文件目录。

```
mkdir -p m3db/config
```

2. 新建m3db/config/m3dbnode.yml文件。

```
vim m3db/config/m3dbnode.yml
```

3. 在m3db/config/m3dbnode.yml文件添加如下配置，

```
        coordinator:
          listenAddress:
            value: "0.0.0.0:7201"
          local:
            namespaces:
              - namespace: default
                type: unaggregated
                retention: 48h
          logging:
            level: info
          metrics:

           scope:
              prefix: "coordinator"
            prometheus:
              handlerPath: /metrics
              listenAddress: 0.0.0.0:7203 # until https://github.com/m3db/m3/issues/682 is
resolved
            sanitization: prometheus
            samplingRate: 1.0
            extended: none
          limits:
            maxComputedDatapoints: 10000
```

```
    tagOptions:
      # Configuration setting for generating metric IDs from tags.
      idScheme: quoted
  db:
    logging:
      level: info
    metrics:
      prometheus:
        handlerPath: /metrics
      sanitization: prometheus
      samplingRate: 1.0
      extended: detailed
    listenAddress: 0.0.0.0:9000
    clusterListenAddress: 0.0.0.0:9001
    httpNodeListenAddress: 0.0.0.0:9002
    httpClusterListenAddress: 0.0.0.0:9003
    debugListenAddress: 0.0.0.0:9004

    hostID:
      resolver: config
      value: m3db_local

    client:
      writeConsistencyLevel: majority
      readConsistencyLevel: unstrict_majority
    gcPercentage: 100

    writeNewSeriesAsync: true
    writeNewSeriesLimitPerSecond: 1048576
    writeNewSeriesBackoffDuration: 2ms
    bootstrap:
      bootstrappers:
          - filesystem
          - commitlog
          - peers
          - uninitialized_topology
      commitlog:
        returnUnfulfilledForCorruptCommitLogFiles: false

    cache:
      series:
        policy: lru
      postingsList:
        size: 262144
    commitlog:
```

```
            flushMaxBytes: 524288
            flushEvery: 1s
            queue:
              calculationType: fixed
              size: 2097152
          fs:
            filePathPrefix: /var/lib/m3db
          config:
              service:
                  env: default_env
                  zone: embedded

                   service: m3db
                  cacheDir: /var/lib/m3kv
                  etcdClusters:
                      - zone: embedded
                        endpoints:
                            - 127.0.0.1:2379
              seedNodes:
                  initialCluster:
                      - hostID: m3db_local
                        endpoint: http://127.0.0.1:2380
          # un-comment the lines below to enable Jaeger tracing. See
https://www.jaegertracing.io/docs/1.9/getting-started/
          # for quick local setup (which this config will send data to).
          # tracing:
          #  backend: jaeger
```

4. 拷贝

```
cp m3db/config/m3dbnode.yml /etc/m3db
```

5. 建立存储数据目录

```
mkdir -p /var/lib/m3db
```

3. 自启动服务

1. 创建systemd 目录

```
mkdir -p m3db/systemd
```

2. 创建m3db/systemd/m3dbnode.service文件

```
vi m3db/systemd/m3dbnode.service
```

3. 在m3db/systemd/m3dbnode.service文件，添加如下配置。

```
[Unit]
Description="M3DB Timeseries Database"
```

```
Documentation=http://m3db.github.io/m3/
After=network.target

[Service]
Type=simple

ExecStart=/usr/local/bin/m3dbnode -f /etc/m3db/m3dbnode.yml
Restart=on-failure
RestartSec=10s
SuccessExitStatus=0

# May not be honored if higher than kernel limit (sysctl fs.file-max) or process
# limit (sysctl fs.nr_open). Also may not be honored if lower than systemd limit
# (system.conf) or systemd user limit (user.conf).
LimitNOFILE=3000000

[Install]
WantedBy=multi-user.target
```

4. 拷贝

```
cp m3db/systemd/m3dbnode.service /usr/lib/systemd/system
```

4. 启动存储节点

```
  systemctl daemon-reload
  systemctl enable m3dbnode
  systemctl start m3dbnode
  systemctl status m3dbnode
```

5. 创建命名空间和初始化拓扑

```
  $ curl -X POST http://localhost:7201/api/v1/database/create -d '{
    "type": "local",
    "namespaceName": "default",
    "retentionTime": "12h"
  }'

  $ curl http://localhost:7201/api/v1/placement | jq .
```

6. 测试验证

```
  $ curl -sS -X POST localhost:9003/writetagged -d '{
    "namespace": "prom_metics",
    "id": "foo",
    "tags": [
      {
        "name": "city",
```

```
        "value": "new_york"
      },
      {
        "name": "endpoint",
        "value": "/request"
      }
    ],
    "datapoint": {
      "timestamp": '"$(date "+%s")"',
      "value": 42.123456789
    }
  }'

  $ curl -sS -X POST http://localhost:9003/query -d '{
    "namespace": "prom_metics",
    "query": {

      "regexp": {
        "field": "city",
        "regexp": ".*"
      }
    },
    "rangeStart": 0,
    "rangeEnd": '"$(date "+%s")"'
  }' | jq .
```

#10.6 安装M3DB集群

129 10.6.1 环境说明

1. 存储节点（包含种子节点）

```
172.24.2.63~65
```

2. 协调器节点

```
172.24.2.62
```

130 11.6.2 安装前的准备

以下操作需要依次在所有节点操作。

1. 关闭防火墙

```
systemctl status firewalld
systemctl stop firewalld
systemctl disable firewalld
systemctl status firewalld
```

2. 使用vi命令编辑/etc/selinux/config文件，关闭SELinux

```
SELINUX=disabled
```

3. 配置时间同步

```
date -s "20200222 21:53:00"
```

4. 内核内置

```
sysctl -w vm.max_map_count=3000000
sysctl -w vm.swappiness=1
sysctl -w fs.file-max=3000000
sysctl -w fs.nr_open=3000000
```

5. 建立缓存目录

```
mkdir -p /var/lib/m3kv
```

6. 安装jq

```
yum install jq
```

131 11.6.3 存储节点安装

可选存储节点任意一台操作，这里选择172.24.2.63 1) 二进制包安装

1. 到https://github.com/m3db/m3/releases网址下载m3_0.14.2_linux_amd64.tar.gz压缩包。

2. 解压m3_0.14.2_linux_amd64.tar.gz

```
tar -xvf m3_0.14.2_linux_amd64.tar.gz
```

3. 执行文件拷贝

```
cp m3_0.14.2_linux_amd64/m3dbnode /usr/local/bin
scp m3_0.14.2_linux_amd64/m3dbnode root@172.24.2.64:/usr/local/bin
scp m3_0.14.2_linux_amd64/m3dbnode root@172.24.2.65:/usr/local/bin
```

2. 配置文件

1. 创建config配置文件目录

```
mkdir -p m3db/config
```

2. 在m3db/config创建m3dbnode.yml配置文件

```
vi m3db/config/m3dbnode.yml
```

3. 在m3dbnode.yml配置文件添加如下配置

```
coordinator:
    listenAddress:
      value: "0.0.0.0:7201"

    local:
      namespaces:
        - namespace: default
          type: unaggregated
          retention: 48h

    logging:
      level: info

    metrics:
      scope:
        prefix: "coordinator"
      prometheus:
        handlerPath: /metrics
        listenAddress: 0.0.0.0:7203 # until https://github.com/m3db/m3/issues/682 is resolved
      sanitization: prometheus
      samplingRate: 1.0
      extended: none
```

```
  tagOptions:
    # Configuration setting for generating metric IDs from tags.
    idScheme: quoted

db:
  logging:
    level: info

  metrics:
    prometheus:
      handlerPath: /metrics
    sanitization: prometheus
    samplingRate: 1.0
    extended: detailed

  hostID:
    resolver: hostname

# Fill-out the following and un-comment before using.
  config:
    service:
      env: default_env
      zone: embedded
      service: m3db
      cacheDir: /var/lib/m3kv
      etcdClusters:
        - zone: embedded
          endpoints:
            - 172.24.2.63:2379
            - 172.24.2.64:2379
            - 172.24.2.65:2379
    seedNodes:
      initialCluster:
        - hostID: chl-servmesh-test1
          endpoint: http://172.24.2.63:2380
        - hostID: chl-servmesh-test2
          endpoint: http://172.24.2.64:2380
       - hostID: chl-servmesh-test3
          endpoint: http://172.24.2.65:2380

  listenAddress: 0.0.0.0:9000
  clusterListenAddress: 0.0.0.0:9001
  httpNodeListenAddress: 0.0.0.0:9002
  httpClusterListenAddress: 0.0.0.0:9003
```

```
    debugListenAddress: 0.0.0.0:9004

    client:
      writeConsistencyLevel: majority
      readConsistencyLevel: unstrict_majority

    gcPercentage: 100

    writeNewSeriesAsync: true
    writeNewSeriesLimitPerSecond: 1048576
    writeNewSeriesBackoffDuration: 2ms

    bootstrap:
      bootstrappers:
          - filesystem
          - commitlog
          - peers
          - uninitialized_topology
      commitlog:
        returnUnfulfilledForCorruptCommitLogFiles: false

    cache:
      series:
        policy: lru
      postingsList:
        size: 262144
  commitlog:
      flushMaxBytes: 524288
      flushEvery: 1s
      queue:
        calculationType: fixed
        size: 2097152

    fs:
      filePathPrefix: /var/lib/m3db
```

4. 拷贝

$ cp m3db/config/m3dbnode.yml /etc/m3db $ scp m3db/config/m3dbnode.yml root@172.24.2.64:/etc/m3db $ scp m3db/config/m3dbnode.yml root@172.24.2.65:/etc/m3db

5. 建立存储数据目录（依次在存储节点操作）

$ mkdir -p /var/lib/m3db

3. 自启动服务

1. 创建systemd 目录

```
mkdir -p m3db/systemd
```

2. 创建m3db/systemd/m3dbnode.service文件

```
vi m3db/systemd/m3dbnode.service
```

3. 在m3db/systemd/m3dbnode.service文件，添加如下配置。

```
$ mkdir -p m3db/systemd

$ vi m3db/systemd/m3dbnode.service
[Unit]
Description="M3DB Timeseries Database"
Documentation=http://m3db.github.io/m3/
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/m3dbnode -f /etc/m3db/m3dbnode.yml
Restart=on-failure
RestartSec=10s
SuccessExitStatus=0

# May not be honored if higher than kernel limit (sysctl fs.file-max) or process
# limit (sysctl fs.nr_open). Also may not be honored if lower than systemd limit
# (system.conf) or systemd user limit (user.conf).
LimitNOFILE=3000000

[Install]
WantedBy=multi-user.target
```

4. 拷贝

```
$ cp m3db/systemd/m3dbnode.service /usr/lib/systemd/system
$ scp m3db/systemd/m3dbnode.service root@172.24.2.64:/usr/lib/systemd/system
$ scp m3db/systemd/m3dbnode.service root@172.24.2.65:/usr/lib/systemd/system
```

4. 启动存储节点 下面操作，需要依次在存储接地那操作。

```
  $ systemctl daemon-reload
  $ systemctl enable m3dbnode
  $ systemctl start m3dbnode
  $ systemctl status m3dbnode
```

5. 创建命名空间和初始化拓扑

```
     $ curl -X POST http://localhost:7201/api/v1/database/create -d '{
    "type": "cluster",
```

```
    "namespaceName": "prom_metics",
    "retentionTime": "48h",
    "numShards": "1024",
    "replicationFactor": "3",
    "hosts": [
          {
              "id": "chl-servmesh-test1",
              "isolationGroup": "us-east1-a",
              "zone": "embedded",
              "weight": 100,
              "address": "172.24.2.63",
              "port": 9000
          },
          {
              "id": "chl-servmesh-test2",
              "isolationGroup": "us-east1-b",
              "zone": "embedded",
              "weight": 100,
              "address": "172.24.2.64",
              "port": 9000
          },
          {
              "id": "chl-servmesh-test3",
          "isolatio
              nGroup": "us-east1-c",
                          "zone": "embedded",
                          "weight": 100,
                          "address": "172.24.2.65",
                          "port": 9000
                      }
                  ]
              }'
              $ curl http://localhost:7201/api/v1/placement | jq .
```

6. 测试验证

```
$ curl -sS -X POST localhost:9003/writetagged -d '{
  "namespace": "prom_metics",
  "id": "foo",
  "tags": [
    {
      "name": "city",
      "value": "new_york"
    },
    {
      "name": "endpoint",
```

```
      "value": "/request"
    }
  ],
  "datapoint": {
    "timestamp": '"$(date "+%s")"',
    "value": 42.123456789
  }
}'

$ curl -sS -X POST http://localhost:9003/query -d '{
  "namespace": "prom_metics",
  "query": {
    "regexp": {
      "field": "city",
      "regexp": ".*"
    }
  },
  "rangeStart": 0,
  "rangeEnd": '"$(date "+%s")"'
}' | jq .
```

132 11.6.4 协调器节点安装

在协调器节点操作，这里是172.24.2.62，一般以Sidecar形式和Prometheus部在一起

1. 二进制包安装

1.解压

```
tar -xvf m3_0.14.2_linux_amd64.tar.gz
```

2.执行文件拷贝

```
cp m3_0.14.2_linux_amd64/m3coordinator /usr/local/bin
```

2. 配置文件

1. 准备

```
mkdir -p m3db/config
```

2. 拷贝

```
mkdir -p /etc/m3coordinator
cp m3db/config/m3coordinator.yml /etc/m3coordinator
```

3. 自启动服务

1. 创建目录

```
mkdir -p m3db/systemd
```

2. 创建m3coordinator.service文件

```
vi m3db/systemd/m3coordinator.service
  [Unit]
  Description="M3 Coordinator"
  Documentation=http://m3db.github.io/m3/
  After=network.target

  [Service]
  Type=simple
  ExecStart=/usr/local/bin/m3coordinator -f /etc/m3coordinator/m3coordinator.yml
  Restart=on-failure
  RestartSec=10s

   SuccessExitStatus=0

  # May not be honored if higher than kernel limit (sysctl fs.file-max) or process
  # limit (sysctl fs.nr_open). Also may not be honored if lower than systemd limit
  # (system.conf) or systemd user limit (user.conf).
  LimitNOFILE=3000000
```

```
[Install]
WantedBy=multi-user.target
```

3. 拷贝

```
cp m3db/systemd/m3coordinator.service /usr/lib/systemd/system
```

4. 启动协调器节点

```
$ systemctl daemon-reload
$ systemctl enable m3coordinator
$ systemctl start m3coordinator
$ systemctl status m3coordinator
```

5. 创建命名空间和初始化拓扑

```
$ curl -X POST http://localhost:7201/api/v1/database/create -d '{
  "type": "cluster",
  "namespaceName": "prom_metics",
  "retentionTime": "48h",
  "numShards": "1024",
  "replicationFactor": "3",
  "hosts": [

 {
    "id": "chl-servmesh-test1",
    "isolationGroup": "us-east1-a",
    "zone": "embedded",
    "weight": 100,
    "address": "172.24.2.63",
    "port": 9000
},
{
    "id": "chl-servmesh-test2",
    "isolationGroup": "us-east1-b",
    "zone": "embedded",
    "weight": 100,
    "address": "172.24.2.64",
    "port": 9000
},
{
    "id": "chl-servmesh-test3",
    "isolatio
    nGroup": "us-east1-c",
                "zone": "embedded",
                "weight": 100,
                "address": "172.24.2.65",
```

```
            "port": 9000
        }
    ]
}'

$ curl http://localhost:7201/api/v1/placement | jq .
```

6. 测试验证

```
$ curl -sS -X POST localhost:9003/writetagged -d '{
  "namespace": "prom_metics",

  "id": "foo",
  "tags": [
    {
      "name": "city",
      "value": "new_york"
    },
    {
      "name": "endpoint",
      "value": "/request"
    }
  ],
  "datapoint": {
    "timestamp": '"$(date "+%s")"',
    "value": 42.123456789
  }
}'

$ curl -sS -X POST http://localhost:9003/query -d '{
  "namespace": "prom_metics",
  "query": {
    "regexp": {
      "field": "city",
      "regexp": ".*"
    }
  },
  "rangeStart": 0,
  "rangeEnd": '"$(date "+%s")"'
}' | jq .
```

133 11.6.5 与Prometheus集成

1. 在prometheus.yml文件中加入如下配置

```
remote_read:

  - url: "http://172.24.2.62:7201/api/v1/prom/remote/read"

  //# To test reading even when local Prometheus has the data

    read_recent: true

   remote_write:

   - url: "http://172.24.2.62:7201/api/v1/prom/remote/write"
```

134 第12章 自动部署AMP使用说明

134.0.1 12.1、部署说明

1. 自动部署只包括AMP产品。不包括mysql、redis、jdk的部署。

134.0.2 12.2、安装

134.1 12.2.1、安装介质

基本介质

- auc-v2.0.tar.gz
- aalarm-manager_SE-v1.2.tar.gz
- amp-components.linux-amd64.zip
- amp-infra-monitor-v3.4.tar.gz
- amp-workorder-v3.3.tar.gz
- apusic-skywalking-apm-bin.tar.gz
- elasticsearch-7.2.0-linux-x86_64.tar.gz

license介质

- license_auc.xml
- license_alarm.xml
- license_infra.xml
- license_workorder.xml
- license_apm.xml

执行脚本介质

- amp_all_install.sh
- amp_install.sh
- apm_install.sh
- restartauc.sh

- `amp_restart.sh`
- `installelasticsearch.sh`

134.2 12.2.2、执行安装（例如安装在/usr/local/amp路径下）

1.将基本介质和执行脚本介质拷贝到/usr/local/amp下，license介质拷贝到/usr/local/amp/license下

2.赋予权限

```
cd /usr/local/amp
```

```
chmod +x amp_all_install.sh
```

3.如果需要修改redis或mysql端口和连接信息，可以编辑amp_install.sh,修改端口和连接信息(不需要修改可以跳过此步骤)

```
vim amp_install.sh
```

4.执行

```
./amp_install_all.sh
```

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