**详细设计说明书**

目 录

[1. 引言 4](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115737#_Toc266115737)

[1.1 编写目的和范围 4](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115738#_Toc266115738)

[1.2 术语表 4](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115739#_Toc266115739)

[1.3 参考资料 4](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115740#_Toc266115740)

[1.4 使用的文字处理和绘图工具 4](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115741#_Toc266115741)

[2. 全局数据结构说明 5](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115742#_Toc266115742)

[2.1 常量 5](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115743#_Toc266115743)

[2.2 变量 5](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115744#_Toc266115744)

[2.3 数据结构 5](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115745#_Toc266115745)

[3. 模块设计 5](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115746#_Toc266115746)

[3.1 用例图 5](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115747#_Toc266115747)

[3.2 功能设计说明 5](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115748#_Toc266115748)

[3.2.1 模块1 5](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115749#_Toc266115749)

[3.2.2 模块2 6](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115750#_Toc266115750)

[4. 接口设计 7](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115751#_Toc266115751)

[4.1 内部接口 7](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115752#_Toc266115752)

[4.2 外部接口 7](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115753#_Toc266115753)

[4.2.1 接口说明 7](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115754#_Toc266115754)

[4.2.2 调用方式 8](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115755#_Toc266115755)

[5. 数据库设计 8](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115756#_Toc266115756)

[6. 系统安全保密设计 8](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115757#_Toc266115757)

[6.1 说明 8](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115758#_Toc266115758)

[6.2 设计 8](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115759#_Toc266115759)

[6.2.1 数据传输部分 8](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115760#_Toc266115760)

[6.2.2 IP过滤分部 8](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115761#_Toc266115761)

[6.2.3 身份验证部分 8](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115762#_Toc266115762)

[7. 系统性能设计 9](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115763#_Toc266115763)

[8. 系统出错处理 9](file:///C:\\Documents%20and%20Settings\\Administrator\\桌面\\详细设计.doc" \l "_Toc266115764#_Toc266115764)

# 引言

## 编写目的和范围

本详细设计说明书编写的目的是说明程序模块的设计考虑，包括程序描述、输入/输出、算法和流程逻辑等，为软件编程和系统维护提供基础。本说明书的预期读者为系统设计人员、软件开发人员、软件测试人员和项目评审人员。

## 术语表

|  |  |  |
| --- | --- | --- |
| **序号** | **术语或缩略语** | **说明性定义** |
| 1 | Nagios | 一个开源的监控框架，简单易扩展。 |
| 2 | Ganglia | 一个开源的监控工具，能够获取大量节点的性能信息 |
| 3 | Graphite | 一个开源的性能数据存储软件。 |
| 4 | Grafana | 一个开源的展示时序图表的仪表盘。 |

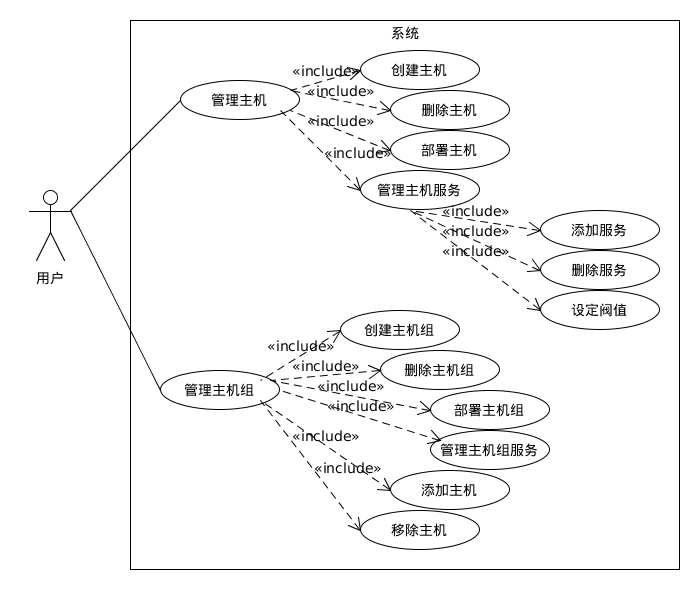
## 参考资料

1. Kevin Jackson, Openstack Cloud Computing Cookbook Third Edition, Packt Publishing Ltd., 2015
2. V. K. Cody Bumgardner, Openstack In Action, Manning Publications Co., 2016
3. David Josephsen，Nagios系统监控实践，机械工业出版社，2014
4. Massie，Ganglia系统监控，机械工业出版社，2013
5. 詹姆斯 特恩布尔,第一本docker书，人民邮电出版社，2014
6. 格林布尔，Flask Web开发：基于Python的Web应用开发实战，人民邮电出版社，2015
7. Lorin Hochstein, Ansible: Up and runing, O’Reilly Media Inc., 2015

Jason Myers&Rick Copeland, Essential SQLAlchemy, O’Reilly Media Inc., 2016

# 模块设计

## 用例图



## 功能设计说明

### 数据库模型

管理系统是一个基于flask框架的web应用，内部使用了sqlalchemy的ORM（对象关系映射）对数据库进行操作。对象关系映射把编程语言的对象映射到关系型数据库的表，对对象的操作会被转化为对数据库表的操作，从而避免了直接写SQL语句。使用对象关系映射提高了开发效率，也避免了直接写SQL语句可能造成的粗心的错误。**[10]**

管理系统没有直接使用sqlalchemy，而是使用flask-sqlalchemy。Flask-sqlalchemy是flask的扩展，它简化了flask应用程序对sqlalchemy的使用，并提供了有用的默认值和辅助函数使得常用的功能用起来更加方便。数据库模型模块内定义了Model类，Model类继承了sqlalchemy的模型类，从而获得映射到关系型数据库的能力。模块还定义了IdMixin和CreateAtMixin用于多重继承。继承IdMixin的模型会拥有id的列作为模型的主键，继承CreateAtMixin的模型会拥有create\_at的列，当插入新的记录时该列会自动被设置为当前时间。模块内的所有其他类都继承了Model类，并成为数据库模型。

类Group是主机组的模型，有以下的方法：

create\_host\_group(name)，用于创建物理机的组，参数name为主机组的名字。

create\_vm\_group(name)，用于创建虚拟机的组。

add\_machine(machine), remove\_machine(machine)，在主机组内添加、移除物理机或虚拟机。

add\_service, remove\_service，为主机组添加、移除服务。

update\_service（services），将主机的服务替换为参数指定的服务。

delete，删除主机组以及主机组相关的服务，并把所有包含的主机移除主机组，如果主机只被当前的主机组包含，则删除掉主机。

类Host是物理机的模型，有以下的方法：

create(ip, hostname, ssh\_user, ssh\_pass），创建一个物理机实例并加入数据库。

get\_all\_standalone\_host，获取所有没有被加入物理机组的物理机。

add\_service, remove\_service，添加、移除物理机的服务。

update\_service(services)，把物理机的服务替换为参数指定的服务。

get\_services，获取物理机关联的所有服务，如果物理机在组中，则返回所有组定义的服务的并集，否则返回物理机定义的服务。

delete，删除物理机，移除所有关联的服务，并从组中删除。

类Vm是虚拟机的模型，有如下方法：

create，创建一个虚拟机实例并加入数据库。

add\_service, remove\_service，update\_service(services)，get\_services，delete的功能类似Host。

### 调用外部系统的模块

要使得用户在管理系统定义的配置真正生效，需要把配置同步到外部系统中。管理系统涉及的外部系统有Ansible、Openstack、Nagios、Grafana，与这些外部系统交互的代码都放在utils文件夹。

Ansible用于在被监控的物理机上部署监控代理，这是个相当消耗时间的操作，所以部署的任务被放到一个单独的线程中执行。模块提供deploy(hosts)的方法供外部调用。调用deploy方法，则Ansible会在hosts定义的所有主机上同时安装和配置gmond守护进程。首先模块会根据hosts参数生成inventory文件，该文件记录了所有要部署的主机的IP地址和SSH账号密码，有了这些信息Ansible才能连接到主机上并拥有操作权限。接着模块会创建一个新的线程，在线程中通过shell命令来启动Ansible任务，并为任务创建一个随机的名字，该名字将被用于保存任务输出的文件的名字。在Ansible任务执行完后，线程获取任务的返回码以判断任务是否成功，并修改数据库中主机的状态。线程还把任务的标准输出保存到以任务名命名的文件中，之后管理系统要查看任务的详细状态时，将读取这个文件，并把输出显示到web页面中。

管理系统会使用Openstack的API获取Openstack创建的所有虚拟机。模块提供了sync的方法供外部使用。调用sync时，模块会从Openstack中获取所有虚拟机的信息，并把虚拟机按照宿主主机分组。模块删除掉数据库中所有虚拟机和虚拟机组的相关定义，并加入新获取的内容。

Nagios模块提供sync()方法把用户的定义同步到Nagios。Nagios的定义包括服务、主机、主机组三个部分，由于定义的格式一致，模块使用了add\_config的方法实现了同步定义的底层实现。add\_config从参数获取定义的类型、键值对和定义要保存的文件，之后根据这些参数生成Nagios的定义并追加写到了文件的末尾。模块内的add\_service, add\_host，add\_group都调用了add\_config方法，并对接口进行了简化。Sync方法做了两件事情，覆盖Nagios定义，重启Nagios服务。有无重启Nagios服务较为消耗时间，为了不影响web应用的性能，将sync方法的主要功能到放入线程中运行。但sync被调用时，就会启动一个线程。在线程中模块清空了Nagios的配置文件，并从数据库中读取用户的定义，分别调用add\_service, add\_host，add\_group方法，把新的定义写入文件中，之后通过shell重启Nagios。

Grafana模块使用sync方法把主机和组同步到仪表盘。模块内仿照Grafana定义了Dashboard、Row、Panel、SingleStat、Graph的类，其中SingleStat和Graph继承了Panel类。一个Dashboard可以包含多个Row，一个Row可以包含多个Panel。SingleStat是简单的面板类型，只显示标题和数值，优点在于可以在有限空间内显示更多的监控项目。Graph则更为常用，可以显示复杂的图表。每个类都提供了add\_children方法用于添加下一级的元素。Sync方法会从数据库读取所有用户定义的主机和组，并为每个主机生成一个详细仪表盘，为每个组生成一个总览仪表盘。

### 提供web服务的模块

该模块属于MVC中的控制层，负责渲染页面和处理用户的请求。用户从浏览器发来的请求被路由到请求的路径对应的函数上，该函数会从请求中解析参数，对参数进行检查，检查成功后执行相应的操作，并返回结果。

该模块内有多个文件。forms.py包含了web应用用到了所有表单。该模块使用了flask-wtf和wtform，简化了web应用对表单的使用。通过这些扩展，web应用可以直接在python代码定义表单的类，利用其解析HTTP请求中的表单并进行验证。此外，还能把表单渲染到前段页面的模板中，减少重复代码。views.py文件包含了与请求路径相互绑定的处理函数，这些函数负责检查和处理用户的请求。

# 数据库设计

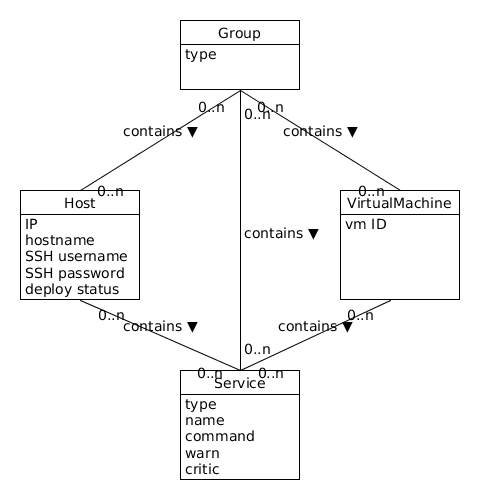


图3-2：领域模型

在监控系统中有两种主机，一种是物理机，其模型为Host，一种是虚拟机，对应模型为VirtualMachine。物理机包含IP、主机名、SSH账号密码的属性。Nagios定义主机对象时，使用了IP、主机名来区分主机，因此两者都必须保持唯一性。SSH账号密码被Ansible自动部署工具用于连接到主机部署监控的代理端。虚拟机包含vm id的属性，该属性是Openstack虚拟机的唯一标识符。主机组的模型为Group，包含类型的属性，用于说明该主机组包含的是物理机还是虚拟机。一个主机组只能包含同一个类型的主机。物理机、虚拟机都可以属于多个组。每个主机、主机组都可以对应多个服务。服务有类型的属性，用于表明该服务属于物理机还是虚拟机。服务的名字是其唯一标识符，不可以重复。服务的命令属性为该服务对应的Nagios命令。服务的警告和严重属性则定义了警告和严重的阀值。

# 系统出错处理

系统中的各种提示如表4-1所示：

表4-1 系统出错提示

|  |  |
| --- | --- |
| 错误类别 | 错误处理 |
| 输入有误 | 提示错误的类型，要求用户重新输入 |
| 同步到其他软件失败 | 提示同步出错 |
| 未知的错误 | 返回500 server error |