目录

[一、环境需求 2](#_Toc437305015)

[ 采集机A： 2](#_Toc437305016)

[ 物理机B： 2](#_Toc437305017)

[ 测试机C： 2](#_Toc437305018)

[二、程序安装 2](#_Toc437305019)

[1. 在A机上安装pip： 2](#_Toc437305020)

[2. 在A机上安装依赖包： 2](#_Toc437305021)

[3. 在A机上配置无密码登陆B物理机： 2](#_Toc437305022)

[4. 建立项目数据库 2](#_Toc437305023)

[5. 在cloud\_config表中添加配置项 2](#_Toc437305024)

[6. 系统配置 3](#_Toc437305025)

[三、测试运行 3](#_Toc437305026)

[1. 在采集机A上运行持续监控后台程序 3](#_Toc437305027)

[2. 在采集机A上运行API程序 3](#_Toc437305028)

[3. 页面展示 3](#_Toc437305029)

[四、API接口 4](#_Toc437305030)

[1. 获取所有虚拟机列表 4](#_Toc437305031)

[2. 设置是否抓取指定虚拟机 4](#_Toc437305032)

[3. 删除指定虚拟机 4](#_Toc437305033)

[4． 按时间起始结束，获取指定虚拟机数据 5](#_Toc437305034)

[5. 设置抓取时间间隔（秒） 6](#_Toc437305035)

[6. 设置获取虚拟机列表间隔（秒） 7](#_Toc437305036)

[7. 添加物理机IP 7](#_Toc437305037)

[8. 删除指定IP的物理机 7](#_Toc437305038)

# 一、环境需求

要求各机器之间可以ping通

* 采集机A：
* Centos 6+
* Python 2.7+
* 物理机B：
* Linux各版本
* Libvirtd
* 测试机C：
* Windows

# 二、程序安装

1. 在A机上安装pip：

安装python2.7+后执行python get-pip.py来安装pip

1. 在A机上安装依赖包：

# yum install python-libvirt

# pip install gevent

# pip install web.py

# pip install mimerender

# pip install mysql-python

1. 在A机上配置无密码登陆B物理机：
2. 在A机下生成公钥/私钥对，按一次回车，它在~/.ssh生成id\_rsa和id\_rsa.pub

# ssh-keygen -t rsa -P ''

1. 把A机下的id\_rsa.pub复制到B机的~/.ssh/目录下，重命名为authorized\_keys

# scp ~/.ssh/id\_rsa.pub root@B机IP:~/.ssh/authorized\_keys

1. 测试无密码登陆：

ssh B机IP 如果不需要输入密码则配置成功

1. 建立项目数据库

在A机数据库中运行cloud\_monitor.sql文件

# cd libvirt\_monitor\_server

# mysql –u root –p

>source sql/cloud\_monitor.sql

1. 在cloud\_config表中添加配置项

|  |  |  |
| --- | --- | --- |
| Key | value | 默认值 |
| interval\_check | 抓取数据间隔（秒） | 200 |
| interval\_travelsal | 连接物理机，获取虚拟机列表间隔（秒） | 200 |
| host | 物理机IP | 无 |

其中host项可添加多条数据，每条数据一个IP

1. 系统配置

在code/cloud\_monitor\_setting.py中填写各项配置，示例如下

# 数据库配置

db\_engine = 'mysql'

db\_server = '数据库IP'

db\_username = '用户名'

db\_password ='密码'

db\_database = 'cloud\_monitor'

# API端口配置

api\_server\_port = 9898

# 三、测试运行

1. 在采集机A上运行持续监控后台程序

# cd libvirt\_monitor\_server

# chmod 755 start-monitor.sh

# ./ start-monitor.sh

监控程序日志将输出到log目录下的daemon.log和monitor.log文件，运行后查看日志，若无错误输出说明运行成功。

1. 在采集机A上运行API程序

# chmod 755 start-api-server.sh

# ./ start-api-server.sh

API程序日志将输出到log目录下得api.log文件，运行后查看日志，若无错误输出说明运行成功。

1. 页面展示

完成上述步骤后在测试机C上

1. 修改html/index.html，第29行，修改为采集机A的IP及端口

var api\_url = "http://173.26.100.211:9898/instances";

1. 修改html/graph.html，第80行，修改为采集机A的IP及端口

var api\_url = "http://173.26.100.211:9898/instances";

1. 在浏览器中打开index.html，可以看到云平台所有虚拟机列表，点击每条最后一项graph链接，进入虚拟机信息图表页面，展示如下说明配置成功。
2. 在graph页面中可选择时间段展示虚拟机信息。



# 四、API接口

以下localhost表示采集机A的IP

1. 获取所有虚拟机列表

|  |  |
| --- | --- |
| http://localhost/instances | |
| HTTP method | GET |
| 返回值 | id: 虚拟机编号  uuid：虚拟机唯一标识  ip：虚拟机所在物理机IP  enable：是否抓取数据,1-是，0-否 |
| 返回值json示例 |  |

1. 设置是否抓取指定虚拟机

|  |  |
| --- | --- |
| http://localhost/enable/虚拟机uuid | |
| HTTP method | POST |
| 参数 | enable：是否抓取数据1-是，0-否 |
| 返回值 |  |
| 返回值json示例 | {'message':'success'} |

1. 删除指定虚拟机

|  |  |
| --- | --- |
| http://localhost/instances/虚拟机uuid | |
| HTTP method | DELETE |
| 参数 |  |
| 返回值 |  |
| 返回值json示例 | {'message':'success'} |

1. 按时间起始结束，获取指定虚拟机数据

|  |  |
| --- | --- |
| http://localhost/ instances /虚拟机uuid | |
| HTTP method | POST |
| 参数 | stime：起始时间  etime：结束时间 |
| 返回值 | time：数据抓取时间（精确到分）  uuid：虚拟机唯一标识  name：虚拟机名称  host：所属物理机IP  state：物理机状态   |  |  | | --- | --- | | 0 | no state | | 1 | the domain is running | | 2 | the domain is blocked on resource | | 3 | the domain is paused by user | | 4 | the domain is being shut down | | 5 | the domain is shut off | | 6 | the domain is crashed | | 7 | the domain is suspended by guest power management | | 8 | NB: this enum value will increase over time as new events are added to the libvirt API. It reflects the last state supported by this version of the libvirt API. |   number\_cpus：cpu数量  cpu\_usage：cpu使用量（%）  max\_memory：最大内存（KB）  memory\_usage：内存使用量（KB）  vir\_interfaces：rx\_packets：下载包数  tx\_packets：上传包数  rx\_bytes：下载数据大小（bytes）  tx\_bytes：上传数据大小（bytes）  rx\_errs：下载错误数  tx\_errs：上传错误数  rx\_drop：下载丢弃  tx\_drop：上传丢弃  vir\_disks： rd\_req：读请求数  wr\_req：写请求数  allocation：已分配  rd\_bytes：读数据大小（bytes）  wr\_bytes：写数据大小（bytes）  errs"：错误数  capacity：硬盘容量  physical：物理磁盘用量 |
| 返回值json示例 | {  "time": "2015-11-14 15:54",  "uuid\_string": "ba16b497-d49d-4a0b-a00b-246209076bfa",  "name": "instance-00000014",  "host": "173.26.100.211",  "state": 1,  "number\_cpus": 1,  "cpu\_usage": "4.667",  "max\_memory": 524288,  "memory\_usage": 524288,  "vir\_interfaces": {  "tap9d6c50ed-03": {  "rx\_packets": 0,  "tx\_packets": 0,  "rx\_bytes": 0,  "tx\_bytes": 0,  "rx\_errs": 0,  "tx\_errs": 0,  "rx\_drop": 0,  "tx\_drop": 0  }  },  "vir\_disks": {  "vda": {  "rd\_req": 0,  "wr\_req": 1.331264531878126,  "allocation": 4595712,  "rd\_bytes": 0,  "wr\_bytes": 2726.429761286402,  "errs": 0,  "capacity": 1073741824,  "physical": 4653056  }  }  } |

1. 设置抓取时间间隔（秒）

|  |  |
| --- | --- |
| http://localhost/interval/check | |
| HTTP method | POST |
| 参数 | interval：时间间隔（秒） |
| 返回值 |  |
| 返回值json示例 | {'message':'success, new check interval is xxx' } |

1. 设置获取虚拟机列表间隔（秒）

|  |  |
| --- | --- |
| http://localhost/interval/travelsal | |
| HTTP method | POST |
| 参数 | interval：时间间隔（秒） |
| 返回值 |  |
| 返回值json示例 | {'message':'success, new check interval is xxx' } |

1. 添加物理机IP

|  |  |
| --- | --- |
| http://localhost/host/add | |
| HTTP method | POST |
| 参数 | host：物理机IP |
| 返回值 |  |
| 返回值json示例 | {'message':'success'} |

1. 删除指定IP的物理机

|  |  |
| --- | --- |
| http://localhost/host/虚拟机IP | |
| HTTP method | POST |
| 参数 |  |
| 返回值 |  |
| 返回值json示例 | {'message':'success'} |

详细调用示例请查看index.html和graph.html