# 一、环境需求

宿主机为linux系统（若有windows系统望提出，进行后续开发）需要安装Docker 任意版本；ovs 2.5.0版本；python3；sqlite3

容器基础镜像需要基于linux系统镜像，若有windows系统镜像望提出，进行后续开发。

（若python版本不同可修改）

需要在宿主机中创建sqlite数据库。

# 二、文件系统

重点包括以下几个文件：

network\_manager.py 文件：运行在manager端，用以记录网络信息及ovs网络的host信息

network\_agent.py 文件：运行在所有的agent端，对容器和OVS网络进行操作，在每台主机中都需要存在。

sqlitemanager.py 文件： 对网络信息进行sqlite数据库进行的操作，运行在manager端。

sqliteagent.py 文件： 对网络配置信息进行sqlite数据库进行的操作，运行在agent端。

其他文件：

randomkey.py 文件：获取随机字符串，用以配置网口名

networkmaindb.db 文件：sqlite数据库文件，记录manager端网络信息。

networkdb.db文件：sqlite数据库文件，记录宿主机上相应网络数据。

# 三、对象及函数相关

## 1.network\_manager.py

network\_manager中可创建Network\_manager对象。其中包含创建swarm overlay网络以及记录ovs网络名称及通信宿主机ip信息。

其中有：

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名 | 参数 | 返回值 |  |
| add\_network | network\_type(str)  user\_name(str)  service\_name(str)  network\_name(str) | 返回网络名或’error’ | 创建网络信息并存储在数据库中，用以控制network\_agent中所创建的网络。 |
| del\_network | network\_type(str)  network\_name(str) | ‘success’或’error’ | 删除数据库中网络信息 |
| add\_swarm\_network | user\_name(str)  service\_name(str)  network\_name(str) | 返回网络名或’error’ | 与add\_network功能一致，单独创建swarm overlay网络信息并存储在数据库中。 |
| del\_swarm\_network | network\_name(str) | ‘success’或’error’ | 删除数据库中swarm overlay网络信息 |
| add\_ovs\_network | user\_name(str)  service\_name(str)  network\_name(str) | 返回网络名或’error’ | 与add\_network功能一致，单独创建ovs网络信息并存储在数据库中。 |
| del\_ovs\_network | network\_name(str) | ‘success’或’error’ | 删除数据库中ovs网络信息 |
| get\_network | network\_type(str)  user\_name(str)  service\_name(str) | 返回网络名或’error’ | 根据用户名和服务名获取数据库中网络名（唯一） |
| add\_host\_ovs\_network | host\_ip(str)  network\_name(str) | ‘success’或’error’ | 用以添加ovs网络中通信主机信息并存储在数据库中。 |
| get\_host\_ovs\_network | network\_name(str) | 存有ip的列表或’error’ | 根据网络名获取ovs网络中所有通信宿主机的ip。 |
| get\_host | host\_ip(str)  network\_name(str) | 存有ip的列表或’error’ | 根据网络名以及宿主机ip判断宿主机ip是否存在相应网络中。 |
| del\_host\_with\_hostip | host\_ip(str)  network\_name(str) | ‘success’或’error’ | 根据宿主机ip及网络名删除相应宿主机ip信息 |
| del\_host\_with\_network | network\_name(str) | ‘success’或’error’ | 根据网络名删除相应宿主机信息 |

### 1)：add\_network

函数名及参数:

add\_network(network\_type,user\_name,service\_name,network\_name)

用以创建网络信息并存储在数据库中。

参数：

network\_type：网络类型，分为’swarm’ 和 ‘ovs’

user\_name：用作用户名

service\_name：服务名，需要通过用户名和服务名确定唯一的网络名

network\_name：用作网络名

返回值：

成功返回网络名，字符串类型，若有异常，则返回“error”开头的异常文本

### 2)：del\_network

函数名及参数:

del\_network(network\_type,network\_name)

用以删除数据库中网络信息。

参数：

network\_type：网络类型，分为’swarm’ 和 ‘ovs’

network\_name：用作网络名

返回值：

成功返回’success’，字符串类型，失败返回’error’，若有异常，则返回“error”开头的异常文本

### 3)：add\_swarm\_network

函数名及参数:

add\_swarm\_network(user\_name,service\_name,network\_name)

用以创建swarm网络信息并存储在数据库中。

参数：

user\_name：用作用户名

service\_name：服务名，需要通过用户名和服务名确定唯一的网络名

network\_name：用作网络名

返回值：

成功返回网络名，字符串类型，若有异常，则返回“error”开头的异常文本

### 4)：del\_swarm\_network

函数名及参数:

del\_swarm\_network(network\_name)

用以删除数据库中swarm网络信息。

参数：

network\_name：用作网络名

返回值：

成功返回’success’，字符串类型，失败返回’error’，若有异常，则返回“error”开头的异常文本

### 5)：add\_ovs\_network

函数名及参数:

add\_ovs\_network(user\_name,service\_name,network\_name)

用以创建ovs网络信息并存储在数据库中。

参数：

user\_name：用作用户名

service\_name：服务名，需要通过用户名和服务名确定唯一的网络名

network\_name：用作网络名

返回值：

成功返回网络名，字符串类型，若有异常，则返回“error”开头的异常文本

### 6)：del\_ovs\_network

函数名及参数:

del\_ovs\_network(network\_name)

用以删除数据库中ovs网络信息。

参数：

network\_name：用作网络名

返回值：

成功返回’success’，字符串类型，失败返回’error’，若有异常，则返回“error”开头的异常文本

### 7)：get\_network

函数名及参数:

get\_network(network\_type,user\_name,service\_name)

获取数据库中网络名。

参数：

network\_type：网络类型，分为’swarm’ 和 ‘ovs’

user\_name：用作用户名

service\_name：服务名，需要通过用户名和服务名确定唯一的网络名

返回值：

成功返回网络名，字符串类型，若有异常，则返回“error”开头的异常文本

### 8)：add\_host\_ovs\_network

函数名及参数:

add\_host\_ovs\_network(host\_ip,network\_name)

用以添加ovs网络中通信主机信息并存储在数据库中。

参数：

host\_ip：主机ip

network\_name：用作网络名

返回值：

成功返回’success’，字符串类型，若有异常，则返回“error”开头的异常文本

### 9)：get\_host\_ovs\_network

函数名及参数:

get\_host\_ovs\_network(network\_name)

根据网络名获取ovs网络中所有通信宿主机的ip。

参数：

network\_name：用作网络名

返回值：

成功返回ip列表，若有异常，则返回“error”开头的异常文本

### 10)：get\_host

函数名及参数:

get\_host(host\_ip,network\_name)

根据网络名以及宿主机ip判断宿主机ip是否存在相应网络中。

参数：

network\_name：用作网络名

返回值：

返回ip列表，若有，则列表长度>0，若无，则列表长度=0

### 11)：del\_host\_with\_hostip

函数名及参数:

del\_host\_with\_hostip(host\_ip,network\_name)

根据宿主机ip及网络名删除相应宿主机信息。

参数：

host\_ip：主机ip

network\_name：用作网络名

返回值：

成功返回’success’，失败返回’error’，若有异常，则返回“error”开头的异常文本

### 12)：del\_host\_with\_network

函数名及参数:

del\_host\_with\_nework(network\_name)

根据网络名删除相应宿主机信息。

参数：

network\_name：用作网络名

返回值：

成功返回’success’，失败返回’error’，若有异常，则返回“error”开头的异常文本

## 2.network\_agent.py

network\_agent中可创建Network\_agent对象。其中包含对容器和OVS网络进行操作的函数。对象无参数。

其中有：

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名 | 参数 | 返回值 |  |
| ovs\_bridge\_create | user\_name(str)  host\_name(str)  vtep\_ip(str)  host\_ips(list) | 返回网桥名或’error’ | 在宿主机上创建OVS网桥，用以替代docker0。 |
| add\_ovs\_network\_remoteip | Br\_name(str)  host\_ips(list) | 无 | 为OVS网桥跨主机vxlan通信设置remote\_ip |
| container\_network\_create | br\_name(str)  container\_id(str)  user\_name(str)  network\_name(str)  container\_ip(str)  tag\_id(int) | 返回容器ip或’error’ | 为容器添加ovs网络，只适用于ovs网络。 |
| container\_get\_portname | container\_id(str) | 返回容器网口名称或’error’ | 根据容器 id获取数据库中存储的与ovs网桥绑定的容器内网口 |
| set\_container\_tag | br\_name(str)  port\_name(str)  container\_id(str)  tag\_id(int) | 返回容器id或’error’ | 配置容器基于ovs网桥的vlan隔离策略 |
| del\_ovs\_bridge\_brname | br\_name(str) | ‘success’或’error’ | 根据ovs网桥名称删除ovs网桥 |
| del\_ovs\_bridge\_hostname | host\_name(str) | ‘success’或’error’ | 根据主机名删除ovs网桥。 |
| del\_container\_network\_pn\_ci | br\_name(str)  port\_name(str)  container\_id(str) | ‘success’或’error’ | 根据容器内网口名及容器id删除容器内网口及网桥端口，既容器与ovs网桥解绑，但并不是删除容器。 |
| del\_container\_network\_containerid | br\_name(str)  container\_id(str) | ‘success’或’error’ | 根据容器id删除容器内网口及网桥端口，使容器与ovs网桥解绑。 |
| del\_container\_network\_nn | br\_name(str)  network\_name(str) | ‘success’或’error’ | 根据网络名删除当前主机下此网络的所有容器内网口及网桥端口，既容器与ovs网桥解绑 |
| get\_ovs\_brname\_hostname | host\_name(str) | 返回网桥名或’error’ | 根据主机名查询主机内是否建立ovs网桥。 |
| container\_swarmnetwork\_connect | container\_id(str)  network\_name(str) | 返回网络名 | 将容器添加到具体网络名的swarm overlay网络中 |
| container\_swarmnetwork\_disconnect | container\_id(str)  network\_name(str) | 返回网路名 | 将容器从具体网络名的swarm overlay网络中移除 |
| container\_drop\_host | host-ips(list) | 无 | 屏蔽docker0网桥对相应主机的访问 |
| network\_nat | network\_card(str)  s\_port(str)  d\_ip\_port(str) | 无 | 设置指定网卡的nat |

### 1)：ovs\_bridge\_create

函数名及参数:

ovs\_bridge\_create(user\_name,host\_name,vtep\_ip,host\_ips)

用以创建ovs网桥。

参数：

user\_name：用作用户名。

host\_name：必要，主机名。

vtep\_ip：必要，用以设置ovs 网桥的ip及网段。

host\_ips：列表，用以标识ovs网络vxlan跨主机通信的remote\_ip。remote\_ip设置后，连接在同一ovs网桥上的容器可以跨主机通信，可传空列表。

返回值：

新建的ovs网桥名称，字符串类型，若有异常，则返回“error”开头的异常文本。创建的网桥名称默认为veth+主机名

### 2)：add\_ovs\_bridge\_remoteip

函数名及参数:

add\_ovs\_bridge\_remoteip(br\_name,host\_ips)

为OVS网桥跨主机vxlan通信设置remote\_ip

参数：

br\_name：ovs网桥名称

host\_ips：列表，可空，用以标识ovs网络vxlan跨主机通信的remote\_ip。remote\_ip设置后，连接在同一ovs网桥上的容器可以跨主机通信

无返回值

### 3)：container\_network\_create

函数名及参数:

container\_network\_create(br\_name,container\_id,user\_name,network\_name,container\_ip,tag\_id)

创建基于ovs的容器网络

参数：

br\_name：连接的ovs网桥名称，同一br\_name上的容器可以互相通信

container\_id：容器id

user\_name：预留，用作用户名

network\_name：预留，用作服务名（或者网络名）

container\_ip：必要，用以设置容器网络的ip及网段。

tag\_id：非必要，可做vlan隔离，也就是隔离机制

返回值：容器ip，字符串类型，若有异常，则返回“error”开头的异常文本

### ：container\_get\_portname(container\_id)

根据容器 id获取数据库中存储的与ovs网桥绑定的容器内网口

参数：

container\_id：容器id

返回值：容器网口名称，字符串类型，若有异常，则返回“error”开头的异常文本

### ：set\_container\_tag

配置容器基于ovs网桥的vlan隔离策略

参数：

br\_name：连接的ovs网桥名称，同一br\_name上的容器可以互相通信

port\_name：容器端口

container\_id：容器id

tag\_id：可做vlan隔离，也就是隔离机制

返回值：容器id，字符串类型，若有异常，则返回“error”开头的异常文本

### ：del\_ovs\_bridge\_brname

函数名及参数：

del\_ovs\_bridge\_brname(br\_name)

根据ovs网桥名称删除ovs网桥。

参数：

br\_name：连接的ovs网桥名称，根据此名称进行ovs网桥的删除

返回值：

成功返回“success”，失败返回“error”开头异常字符串

### ：del\_ovs\_bridge\_hostname

函数名及参数：

del\_ovs\_bridge\_hostname(host\_name)

根据主机名删除ovs网桥。

参数：

network\_name：服务名（网络名）

返回值：

成功返回“success”，失败返回“error”开头异常字符串

### ：del\_container\_network\_pn\_ci

函数名及参数：

del\_container\_network\_pn\_cd(port\_name,container\_id)

根据容器内网口名及容器id删除容器内网口及网桥端口，既容器与ovs网桥解绑，但并不是删除容器。

参数：

br\_name：ovs网桥名

port\_name：容器端口名

container\_id：容器id

返回值：

成功返回“success”，失败返回“error”开头异常字符串

### ：del\_container\_network\_containerid

函数名及参数：

del\_container\_network\_containerid(container\_id)

根据容器id删除容器内网口及网桥端口，使容器与ovs网桥解绑。

参数：

br\_name：ovs网桥名

container\_id：容器id

返回值：

成功返回“success”，失败返回“error”开头异常字符串

### ：del\_container\_network\_nn

函数名及参数：

del\_container\_network\_nn(network\_name)

根据网络名删除当前主机下此网络的所有容器内网口及网桥端口，既容器与ovs网桥解绑

参数：

br\_name：ovs网桥名

network\_name：服务名（网络名）

返回值：

成功返回“success”，失败返回“error”开头异常字符串

### ：get\_ovs\_brname\_hostname

函数名及参数：

get\_ovs\_brname\_hostname(host\_name)

根据主机名查询主机内是否建立ovs网桥。

参数：

host\_name：主机名

返回值：

成功返回网桥名，若无网桥则返回空字符串，失败返回“error”开头异常字符串

### ：container\_swarmnetwork\_connect

函数名及参数：

container\_swarmnetwork\_connect(container\_id,network\_name)

将容器添加到具体网络名的swarm overlay网络中。

参数：

network\_name：网络名

返回值：

返回网络名

### ：container\_swarmnetwork\_disconnect

函数名及参数：

container\_swarmnetwork\_disconnect(container\_id,network\_name)

将容器从具体网络名的swarm overlay网络中移除。

参数：

network\_name：网络名

返回值：

返回网络名

### ：container\_drop\_host

函数名及参数

container\_drop\_host(host\_ips)

屏蔽docker0网桥对相应主机的访问。

由于容器要与外界访问，需要连接在docker0网桥上，同时，在宿主机内，docker0网桥会创建DOCKER的iptables规则，因此需要对此规则进行drop限制。

参数：

host\_ips：需要屏蔽的主机ip

返回值：

无

### ：container\_remove\_drop\_host

函数名及参数

container\_remove\_drop\_host(host\_ips)

解除docker0网桥对相应主机的屏蔽。

参数：

host\_ips：需要解除屏蔽的主机ip

返回值：

无

### ：network\_nat

函数名及参数

network\_nat(network\_card,s\_port,d\_ip\_port)

设置指定网卡的nat。

参数：

network\_card：当前主机指定网卡

s\_port：当前主机转发源端口

d\_ip\_port：目的主机及端口

返回值：

无

# 四、代码说明

## 1.管理端和代理

Network\_manager运行在manager端中，不管是ovs还是swarm overlay网络，都需要先在network\_manager中配置好网络名称。

Network\_agent运行在所有agent中，为代理所在宿主机中的容器配置相应网络时使用。

## 2.创建ovs网络环境

使用ovs网络环境可使容器间互相通信或跨主机通信，修改过后的架构是参考swarm overlay网络，所有容器共用同一ovs网桥，通过ip网段进行容器的隔离。需要在为容器配置网络前，在主机中建立ovs网桥。用以取代docker0。

创建ovs网络环境首先需要搭建ovs网桥，调用ovs\_network\_create接口即可在宿主机内创建ovs网桥。

示例：

networkagent.ovs\_bridge\_create('tjx1','host1','200.0.0.1/8',[])

处在同一网段上的容器可以互相通信（注：网桥ip必须设置为容器ip的上层网段）。

例如：

Ovs网桥ip为200.0.0.1/8

容器A的ip为200.0.1.1/24

容器B的ip为200.0.1.2/24

容器C的ip为200.0.2.1/24

容器D的ip为200.0.2.2/24

则容器A和B可以通信，C和D可以通信。A与C、D或B与C、D无法通信。

OVS网桥可不设置IP，但若有容器使用bridge模式链接在docker0网桥上，则默认的路由规则是在docker0的网段上，无法与ovs网桥通信。因此ovs网桥需要设置ip。

## 3.创建容器网络环境

### 1）：与外部完全隔离的容器网络配置

需要在容器创建时设置相应参数，包括：

--net=none

设置容器网络模型为none，用以搭建自己的ovs网络

--privileged=true

此参数设置为true意为可以修改容器内部虚拟网口及ip，为必要参数，否则无法为其设置虚拟网口及ip

创建容器后，调用container\_network\_create接口即可创建容器网络。

示例：

networkagent.container\_network\_create('vethservice1','e1985b76cf46e7784fe9400981856e383a50a561c293c256321f391aa8c3d46f','tjx','service1','200.1.2.3/24',100)

其中，第一个参数br\_name为容器需要绑定的ovs网桥的名称，若不知道网桥名称，则可通过get\_ovs\_brname\_hostname接口通过主机名进行获取

示例：

networkagent.get\_ovs\_brname\_hostname('host1')

创建网络环境后，容器与外部网络完全隔离，只能与相应网络中的其他容器进行通信。

### 2）：可访问外部的容器网络配置

需要在容器创建时设置相应参数，包括：

--net=bridge

设置容器网络模型为bridge，表明容器有一网口绑定在docker0网桥上，使用此方法可使容器访问外部时使用NAT方式进行通信。

--privileged=true

创建容器后，调用container\_network\_create接口即可创建容器网络。

此时容器内部有3个网口，一个是lo网口，一个是与docker0网桥对端的网口，一个是与ovs网桥对端的网口。这种情况下，容器即可以访问外部网络，也可以在相应网络中与其他容器进行通信。

### 3）：可被外部访问的容器网络配置

需要在容器创建时设置相应参数，包括：

--net=bridge

设置容器网络模型为bridge，表明容器有一网口绑定在docker0网桥上，使用此方法可使容器访问外部时使用NAT方式进行通信。

--privileged=true

-p 8080:8080

设置容器与宿主机的端口映射，其中“:”前的端口号为主机开放端口，“:”后的端口号为容器内部可提供访问的端口。

创建容器后，调用container\_network\_create接口即可创建容器网络。

由于设置了端口映射关系，外部网络访问容器只需访问宿主机相应ip及端口即可。

### 4）：跨主机容器网络通信

Ovs网络环境提供vxlan的跨主机通信协议，在需要相互通信的两台宿主机的ovs网桥中建立remote对应关系。

在初始化宿主机ovs网桥时使用ovs\_network\_create接口传递最后一个参数，参数为其他宿主机ip列表。

也可以使用add\_ovs\_network\_remoteip接口

示例：

有三台主机，

主机A的ip为：192.168.0.1

主机B的ip为192.168.0.2

主机C的ip为192.168.0.3

若主机A已经搭建了ovs网桥，

则在主机B中创建同一网络的ovs网桥时传递参数[‘192.168.0.1’]，同时主机A中使用add\_ovs\_network\_remoteip传递参数[‘192.168.0.2’]

之后在主机C中创建同一网络的ovs网桥时，传递参数[‘192.168.0.1’,’192.168.0.2’]，同时在主机A和主机B中使用add\_ovs\_network\_remoteip传递参数[‘192.168.0.3’]

注：add\_ovs\_network\_remoteip(br\_name,host\_ips)的第一个参数需要对应的ovs网桥名。若不知道网桥名，可通过get\_ovs\_brname\_hostname接口获取。

## 4.删除某OVS网络下的所有容器

若有服务结束，则在关闭此服务下所有容器之外还需将此服务下所有容器的网络信息删除。

可在服务端通过get\_host\_ovs\_network(network\_name)获取此服务网络下的主机（这一步如果知道具体主机可省略），然后在相应主机agent内通过del\_container\_network\_nn(network\_name)删除所有容器网络信息，包括数据库信息及ovs网桥的相应端口。注：需要运行在删除容器之前。

# 使用说明

## 1：容器ovs网络创建流程

整体流程如下图所示：



### 1）：与创建network网络流程

在管理端调用创建网络的接口，在接口中添加add\_network或者add\_ovs\_network接口，即可将网络信息存储在数据库中，作为网络状态全局配置信息。

### 2）：创建容器流程

在相应宿主机的agent中，如果获取到网桥名，则首先运行docker run命令创建相应容器。之后调用network\_agent中的container\_network\_create接口创建网络。并可以选择调用set\_cntainer\_tag接口为容器设置vlan隔离。可选择调用container\_drop\_host及network\_nat为容器及docker网桥设置iptables规则。

### 3）：与创建ovs网桥流程

根据网络创建的接口判断网络信息是否存储在数据库中，若成功，则通过编排调度确定需要创建容器的宿主机，在创建容器之前，需要先在宿主机中创建ovs网桥。通过调用相应宿主机的agent调用创建网桥的接口，首先在agent相应接口中调用network\_agent中get\_ovs\_brname\_hostname判断宿主机中是否已有网桥，若有，则返回网桥名，若没有，则调用ovs\_bridge\_create创建网桥并返回网桥名，并可以选择调用add\_ovs\_bridge\_remoteip来设置网桥联通的宿主机。

## 2：容器ovs网络删除流程

整体流程如下图所示：



在管理端若需要删除某一容器，需要与相应宿主机Agent通信后调用删除容器接口，并调用del\_container\_network\_\*类别的接口，分别有：del\_container\_network\_pn\_ci接口，del\_container\_network\_containerid接口，del\_container\_network\_nn接口，这些接口只将容器与ovs网桥解绑并从数据库删除数据，并为真正删除容器，还需要在宿主机agent中执行docker stop及docker rm命令。之后可由Agent管理员判断是否删除网桥，调用del\_ovs\_bridge\_\*类别接口，分别有del\_ovs\_bridge\_brname接口、del\_ovs\_bridge\_hostname接口。此类接口会删除ovs网桥以及数据库信息。

注：删除ovs网桥不必要使用，因为ovs网桥可以一直存在宿主机中，同时若ovs网桥删除后，若本身之前还有容器链接，则链接断开。

# 六、部分问题及需求

## 1.iptables限制时传掩码

在container\_drop\_host中，其参数描述为宿主机ip，其实这参数可以当做ip段，因为在iptables的命令规则中，可直接通过ip网段进行屏蔽，例如传参“100.1.1.1”代表限制100.0.0.1的主机流量传输，同时也可以传参“100.1.1.1/24”，代表限制100.1.1.255这个掩码下的所有ip网段。

## 2.使用进程进行检测功能

由于需要使用python进程功能进行开发，并且需要进行测试及验证，因此暂时未开发完成，

注：此功能需要双方确定调用方式及参数传递规则。

# 七、数据库

### 管理端

CREATE TABLE NETWORKTABLE \

(ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,\

NETWORKTYPE TEXT NOT NULL,\

USERNAME TEXT,\

SERVICENAME TEXT NOT NULL,\

NETWORKNAME TEXT NOT NULL);

CREATE TABLE HOSTTABLE \

(ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,\

NETWORKNAME TEXT NOT NULL, \

HOSTIP TEXT NOT NULL,\

SERVICENAME TEXT);

### Agent端

CREATE TABLE OVSTABLE \

(ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,\

BRNAME TEXT NOT NULL,\

USERNAME TEXT,\

SERVICENAME TEXT NOT NULL,\

VTEPIP TEXT NOT NULL);

CREATE TABLE CONTAINERTABLE \

(ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,\

CONTAINERID TEXT NOT NULL, \

BRNAME TEXT NOT NULL,\

PORTNAME TEXT NOT NULL,\

USERNAME TEXT,\

SERVICENAME TEXT,\

CONTAINERIP TEXT NOT NULL);

# 八、异常管理

## 服务端异常

1)error:1-databse error

数据库错误

1. error:2-network name is using

网络名 已使用

1. error:3-network not found

网络没有找到

1. error:4-host in the network

主机已存在网络中

## 客户端异常

1)error:1-databse error

数据库错误

1. error:2-bridge is using

主机中已有虚拟网桥

1. error:3-container is connect bridge

容器已连接在网桥上

1. error:4-container is not connect bridge

容器未连接在网桥上

1. error:5-bridge is not found

主机中未发现网桥

1. error:4-container is not connect bridge

容器未连接在网桥上

1. error:4-container is not connect bridge

容器未连接在网桥上