Technical document on ConPaaS services: ConPaaS MySQL Server

Aleš Černivec

December 19, 2011

Contents

1 Introduction

Currently you can add and remove agent nodes, query for status of the agent nodes, configure users, upload mysql database.

2 Architecture

ConPaaS MySQL Manager node has to be run manually or by the usage of ConPaaS web front-end. ConPaaS Servers can also be managed through direct web front-end (see section 4).

This is only for development: when manager starts, it fetches the fresh package of conpaas sources from public location (e.g. `http://contrail.xlab.si/conpaassql.tar`). Images will be pre-packaged with mysql manager and agent on the release. Template for creating manager, contains details on installation script (`http://contrail.xlab.si/conpaassql/manager/conpaas-install.sh`, compare to section 3). When installation is complete, manager can be used to orchestrate ConPaaS SQL agents.

When obtaining access point of the manager, new agents can be provisioned by issuing HTTP POST `add_nodes` command on resource `sql-manager-host:/`:

```
POST / HTTP/1.1
Accept: */*
Content-Type: application/json
Content-Length: 67
{"params": {"function": "agent"}, "method": "add_nodes", "id": "1"}
```

Parameter `function` will soon support more than just creating new agent nodes (e.g. cluster manager, cluster agent, cluster). Parameter `method` designates command `add_nodes` and `id` equals 1.

3 Installation on VM images

These steps are necessary in order to clean install ConPaaSSQL server on images used on OpenNebula.

First, you will need

1. `http://contrail.xlab.si/conpaassql/agent/conpaas-install.sh`
2. `http://contrail.xlab.si/conpaassql/manager/conpaas-install.sh`
3. `http://contrail.xlab.si/conpaassql.tar` This is a package from SVN.

step1

Copy

- (1) anywhere on ONE manager node, e.g. `root@onehead:/home/contrail/agent/conpaas-install.sh` (4)
- (2) anywhere on ONE manager node (different than (4), e.g. `root@onehead:/home/contrail/manager/conpaas-install.sh` (5))

You will need this for the contextualization process.

step2

Download (3), untar it somewhere for editing (e.g. under `root@onehead:/home/contrail/temp/conpaassql-temp` (6)).

step3

`cd` to (6), change `./src/conpaas/mysql/server/agent/configuration.cnf`:

```
[MySQL_root_connection]
password= [mysql user's password]
username=[mysql username]
```

step4

`cd` to (6), change `./src/conpaas/mysql/server/manager/configuration.cnf` in a following name (substitute IPs, Image ID, Network ID, paths to agent and manager install scripts):

```
OPENNEBULA_URL=http://10.30.1.1:2633/RPC2 # your ONE installation
OPENNEBULA_IMAGE_ID=193 # image of mysql manager on ONE
OPENNEBULA_NETWORK_ID=205 # working network on ONE
OPENNEBULA_CONTEXT_SCRIPT_MANAGER=[location of (5) on ONE]
OPENNEBULA_CONTEXT_SCRIPT_AGENT=[location of (4) on ONE]
```

step5

`tar` the content under (6) again to `conpaassql.tar` somewhere where VMs running on ONE can `wget` from. (see also step 6).

step6

change agent install script (1) in a following way:

```
SERVER=contrail.xlab.si                       # public location, somewhere that VMs on O
PACKAGE_NAME=conpaassql.tar                 # package from step 5)
DEST_DIR=/home/contrail/conpaassql         # location on agent VM
```

step7

change manager install script (2) in a following way:

```
SERVER=contrail.xlab.si # public location, somewhere that VMs on ONE can wget from
PACKAGE_NAME=conpaassql.tar # package from step 5)
DEST_DIR=/home/contrail/conpaassql # location on manager VM
```

For deploying ConPaaS SQL Server image, we are using this template description:

```
NAME   = conpaassql-manager
CPU    = 0.2
MEMORY = 512
   OS     = [
   arch = "i686",
   boot = "hd",
   root     = "hda" ]
DISK   = [
   image_id = "193", // The same as in step 4
   bus = "scsi",
   readonly = "no" ]
NIC    = [ NETWORK_ID = 205 ] // The same as in step 4
GRAPHICS = [
  type="vnc"
  ]
CONTEXT = [
  target=sdc,
  files = /home/ales/sql/manager/conpaassql-install.sh \
  // the same as in step 1, location (5)
  ]
RANK = "- RUNNING_VMS"
```

4 ConPaaS MySQL Server Web front-end

ConPaaS MySQL Server Web front-end can easily be installed after the server already runs:

```
# apt-get install python-setuptools python-dev python-pycurl -y
# easy_install pip
# pip install oca apache-libcloud
# pip install --extra-index-url http://eggs.contrail.xlab.si \
conpaassql-manager-gui
# mkdir -p /etc/conpaassql
# cat > /etc/conpaassql/manager-gui.conf << EOF
# MANAGER_HOST = 'localhost'
# EOF
# screen -S conpaasssql-manager-gui -d -m conpaassql_manager_gui
```

It also is not mandatory that the web front-end is installed on the same server as the SQL Server already runs.

5 ConPaaS MySQL Server Manager API

Module `conpaas.mysql.server.manager.internals` contains internals of the ConPaaS MySQL Server. ConPaaS MySQL Server consists of several nodes with different roles.

- Manager node
- Agent node(s)
  - Master
  - Slave(s)

platform: Linux, Debian Squeeze, tested also within Ubuntu 10.10 (there should be no problem when using later distributions).
synopsis: Internals of ConPaaS MySQL Servers.
moduleauthor: Ales Cernivec <ales.cernivec@xlab.si>

`conpaas.mysql.server.manager.internals.add_nodes(kwargs)`

Description:

HTTP POST method. Creates new node and adds it to the list of existing nodes in the manager. A role of new node can be one of: agent, manager. Currently only agent is supported. It makes internal call to `createServiceNodeThread()`.

Parameters:

kwargs – string describing a function (agent).

Returns:

HttpJsonResponse - JSON response with details about the node.

Raises:

ManagerException

Example

```
POST / HTTP/1.1
Accept: */*
Content-Type: application/json

Body content: {"params": {"function": "agent"}, "method": /
"add_nodes", "id": "1"}
```

`conpaas.mysql.server.manager.internals.remove_nodes(params)`

Description:

HTTP POST method. Deletes specific node from a pool of agent nodes. Node deleted is given by `{'serviceNodeId':id}`.

Parameters:

kwargs –string identifying a node.

Returns:

HttpJsonResponse - HttpJsonResponse - JSON response with details about the node. OK if everything went well.

Raises:

ManagerException if something went wrong. It contains a detailed description about the error.

Example

```
POST / HTTP/1.1
Accept: */*
Content-Type: application/json

Body content: {"params": {"serviceNodeId": "12"}, "method": /
"remove_nodes", "id": "1"}
```

`conpaas.mysql.server.manager.internals.list_nodes()`

Description:

HTTP GET method. Uses `IaaSClient.listVMs()` to get list of all service nodes. For each service node it checks if it is in servers list. If some of them are missing they are removed from the list. Returns list of all service nodes.

Parameters:

Returns:

HttpJsonResponse - JSON response with the list of services: `{ 'serviceNode': [<a list of ids>]})`

Raises:

HttpErrorResponse

Example

```
GET /?method=list_nodes&id=1 HTTP/1.1
Accept: */*
Content-Type: application/json
```

`conpaas.mysql.server.manager.internals.get_node_info()`

Description:

HTTP GET method. Gets info of a specific node.

Parameters:

param (str) – serviceNodeId is a VMID of an existing service node.

Returns:

HttpJsonResponse - JSON response with details about the node: : `{'serviceNode':{'id': serviceNode.vmid,'ip': serviceNode.ip,'isRunningMySQL': serviceNode.isRunningMySQL}}`.

Raises:

ManagerException

Example

```
GET /?params=%7B%22serviceNodeId%22%3A+%221%22%7D&method=get_node_info&id=1 HTTP/1.1
Accept: */*
Content-Type: application/json
```

`conpaas.mysql.server.manager.internals.get_service_info()`

Description:

HTTP GET method. Returns the current state of the manager.

Parameters:

param (str) – serviceNodeId is a VMID of an existing service node.

Returns:

HttpJsonResponse - JSON response with the description of the state.

Raises:

ManagerException

Example

```
GET /?method=get_service_info&id=1 HTTP/1.1
Accept: */*
Content-Type: application/json
```

`conpaas.mysql.server.manager.internals.set_up_replica_master()`

Description:

HTTP POST method. Sets up a replica master node.

Parameters:

id – new replica master id.

Returns:

HttpJsonResponse - JSON response with details about the new node. ManagerException if something went wrong.

Raises:

ManagerException

`conpaas.mysql.server.manager.internals.set_up_replica_slave()`

Description:

HTTP POST method. Sets up a replica master node.

Parameters:

id – new replica slave id.

Returns:

HttpJsonResponse - JSON response with details about the new node. ManagerException if something went wrong.
Raises:
ManagerException

`conpaas.mysql.server.manager.internals.shutdown()`

Description:
HTTP POST method. Shuts down the manager service.
Parameters:
id – new replica slave id.
Returns:
HttpJsonResponse - JSON response with details about the status of a manager node: :py:attr‘S_EPILOGUE‘. ManagerException if something went wrong.
Raises:
ManagerException

`conpaas.mysql.server.manager.internals.get_service_performance()`

Description:
HTTP GET method. Placeholder for obtaining performance metrics.
Parameters:
kwargs (dict) – Additional parameters.
Returns:
HttpJsonResponse – returns metrics
Raises:

Example

```
GET /?method=get_service_performance&id=1 HTTP/1.1
Accept: */*
Content-Type: application/json
```

6 Conclusion

References