Tribble a unified Cluster Management API

Date: 2013-05-01 14:22
tags: rackspace, openstack, aws, ec2, amazon, cloud, clustering, api
category: *nix

Open Tribble Cloud Clustering API

Tribble is an API system with allows for multiple cloud deployments, on multiple cloud providers all based on schematics. The system uses an authenticated system which restricts access to resources based on a username, key, and password. Authentication is presently done through the API and managed via the "cloudauth" table. Usernames and Passwords are stored in this table which restrict access to the various API endpoints and the resources within the system.

Schematic Definition

A *Schematic* contains the cloud provider information, pertinent information on configuration management, and zones.

Zones Definition

A *Zone* contains the initial run script used for configuration management, if any is provided. A Zone also contains information needed to construct instances and a reference to the keys used for the instance upon boot.

Instances Definition

A *Instance* is the embodiment of a Cloud Server

Headers used to access the API resources

X-User: SomeUserName
X-Secretkey: key
X-Password: password

Available Endpoints

- https://localhost:5150/v1/schematics
- https://localhost:5150/v1/schematics/<schematic_id>
- https://localhost:5150/v1/schematics/<schematic_id>/zones
- https://localhost:5150/v1/schematics/<schematic_id>/zones/<zones_id>

Example usage for using GET with the API

```
curl -X GET --insecure -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics
curl -X GET --insecure -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics/<schematic_id>
curl -X GET --insecure -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics/<schematic_id>/zones
curl -X GET --insecure -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics/<schematic_id>/zones/<zone_id>
```

Example usage for using PUT with the API

```
curl -X PUT --insecure -d '{"cloud_region": "dfw"}' -H "Content-Type: application/json" -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics/<schematic_id>
curl -X PUT --insecure -d '{"quantity": 25}' -H "Content-Type: application/json" -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics/<schematic_id>/zones/<zone_id>
```

Example usage for using DELETE with the API

```
curl -X DELETE --insecure -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics/<schematic_id>
curl -X DELETE --insecure -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics/<schematic_id>/zones/<zone_id>
```

Example usage for using POST with the API

```
curl -X POST --insecure -T /path/to/json.file -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics
curl -X POST --insecure -T /path/to/json.file -H "x-user: someusername" -H "x-secretkey: keyiused" -H "x-password: password" https://localhost:5150/v1/schematics/<schematic_id>/zones
```

Please see the example directory for an example of a valid JSON schematic with all available options. *NOTE* not all options are needed or other wise required.

WARNINGS

- In this generation of the application there is NO user management via the API. User management is done all through the administration client which is only available on the local box where the application is installed.
- This is very much in development. Expect changes in the API, as well as table / column layout.
- Shoot me a line if you have any questions.
- NOT everything is working with 100% efficiency.
- NOT everything is working as intended, IE: If you "*PUT*" any updates to the system, they will be received and reflected in the database, however nothing is done with the updated information. The only way the system will take action in an environment is through a "*POST*" or a "*DELETE*". This will be changing soon, though is worth noting now.
- This is NOT production ready yet. But will be eventually.

If you would like to help out please send in your pull requests

Installation

1. Setup a MySQL Database, somewhere.
2. Login to said MySQL and create the database you want to use for the system, also create a user to access that database.
3. Go back to your system where you are going to be running the application. Create the directory "/etc/Tribble", then create the file "config.cfg" in that directory and set the permissions to "*600*". Add the following variables to the config file.

```
[basic]
log_level = info
DB_USERNAME = mysqlusername
DB_PASSWORD = mysqlpassword
DB_HOST = mysqlhostaddress
DB_PORT = 3306
DB_NAME = mysqldatabasename
DB_ENGINE = mysql
debug_mode = True
```

4. Use `PIP` to install "apache-libcloud", "fabric", "python-daemon==1.6", "SQLAlchemy", "Flask-RESTful", "Flask", and "Flask-SQLAlchemy"

5. Go to the bin directory in the cloned application
6. Run the file "dpc_dbcreate.py" (This will create the needed tables)
7. Run the file "dpc_admin.py" (This will create you a user)
8. Run the file "dpc_keycreate.py" (This will create your Self Signed SSL)