Cloudmesh client extension for AWS

GREGOR VON LASZEWSKI[1], MILIND SURYAWANSHI[1], AND PIYUSH RAI[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.

project-006, May 7, 2017

ASW client is an extension to the Cloudmesh client [1] for ineteracting with Amazon Web Services. Cloudmesh client is command line tool to access numerous cloud envirnments and manage the deployment of virtual machines on them. It has it's own command shell [2]. AWS client extends its capability to work with Amazoz EC2 and is based on Libcloud EC2 driver [3].

Keywords: Cloud, I524

Report: https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IR-P006/report/report.pdf
Code: https://github.com/cloudmesh/cloudmesh.aws

INTRODUCTION

AWS provides a cloud service to deploy virtual machines (VM) with numerous operating systems environment availble in different flavors. The images are called Amazon Machine Image (AMI) and are available as preconfigured for different applications. Once can also create his own custom environment [4].

The Amazon EC2 drivers provides numerous functionalities to authenticate into the aws, list available configurations and create and boot VMs. The AWS Client is based on cloudmesh.cmd5 and cloudmesh.common. It uses mongodb in the backend to store the cloud information such as list of availble images or instances running on the cloud. Requests library [5] is used to connect to the backend through rest services. The rest service is deployed using cloudmesh REST framework [6].

GETTING STARTED

One needs to create an aws account first to be able to acces its cloud. The instructions for it are available at [7]. A pair of access key and secret keys are required to be generated to authenticate into the cloud [8]. These keys are required to be kept confidential. These keys are required to be specified in a yaml configuration file. Default vm image and flavor can also be specified in the configuration file. Following are some of the configuration entries:

```
cloudmesh:
  clouds:
    aws:
      credentials:
        EC2_ACCESS_KEY: 'ACCESS KEY'
        EC2_SECRET_KEY: 'SECRET KEY'
        .
        .
default:
  image: 'IMAGE ID'
  size: 'SIZE ID'
  location: 'LOCATION'
```

The user need to ensure that the combination of image, size and location is valid and that it's account has the required priviliges for it. The instructions for installation of AWS client can be found at [9]. Once, the client has been installed and configurations settings enabled, the mongodb and the rest servies to access the database can be started by following command:

```
cms admin mongodb start
cms admin rest start
```

The above services will be requried by AWS client to store cloud related information locally. The user can now execute the client commands e.g.:

```
cms aws flavor refresh
```

This will fetch the list of image sizes available on Amazon EC2 cloud.

ARCHITECTURE

The commands are implemented as methods of a class AwsCommand which is based on PluginCommand from cloudmesh shell. Depending on the arguments passed, corresponding routine is called from Aws client API which acts as a wrppaer around the libcloud EC2 drivers and is also responsible to connect with backend database apart from reading the configuratio from yaml file. The entire code is written in Python.

Technologies Used

1. Cloudmesh.common and Cloudmesh.cmd5: The common library contains number of modules such as for printing and measuring execution time. The cmd5 is an "dynamically extensible CMD based command shell" [2].

2. Libcloud EC2 driver: The driver provides a number of functions for various functionalities such as from listing the available nodes to generating a key pair and deplying a VM.

3. MongoDb: MongoDB is an open source document store database. It's used by AWS client to store information about various VM configuration options available on the Amazon EC2 cloud. It's also used to store information regarding the VMs that are running on the cloud. The information in the database is refreshed whenever it's fetched from the cloud.

4. Cloudmesh.rest: The cloudmesh rest framework is used to deploy the start the mongodb and rest services. The schema for the objects to be stored in the database is collectively specified in json file called 'all.json'. The schema defined in the file is closely associated with the code in awsclient.py responsible for fetching the information from cloud and passing it to mongodb through rest services. From our expericence during the development of this project, we observed that rest services required the schema to be precise and didn't handle null values for collection fields. There's another file all.settings.py which contains the configuration information for rest services such as port mongodb is running on alongwith the database name. It contains the schema for the collections and the list of methods to be provided by the rest services. The database connectivity was intially developed using pymongo library. However, during the review it was suggested that the code based on pymongo is quite low level and does not have adequate security features. This led to the use of Pyhthon Requests library.

AWS COMMANDS

1. Refresh: Whether to always fetch the information from the cloud over the network or display it from the local database when asked can be configured by setting the configuration variable 'refresh' to either 'on' or 'off'. When the value is set to on, the onformation will always be fetched from the cloud.

```
cms aws refresh on
```

2. Image refresh: The images available on the cloud could be listed using this command. The databse will also be updated with the newly fetched list.

```
cms aws image refresh
```

3. Image list: Depending on whether the value of 'refresh' is set to 'on' or 'off', the list is either fetched from the cloud or from the local database.

```
cms aws image list
```

4. Flavor refresh: This will list the different sizes of VMs that are available on the cloud. The information is fetched from the cloud and stored locally.

```
cms aws flavor refresh
```

5. Flavor list: The flavor list is either fetched from the cloud or from the local database depending on whether the value of 'refresh' is set to 'on' or 'off'.

```
cms aws flavor list
```

ACKNOWLEDGEMENTS

Prof. Gregor von Laszewski originally suggested this project and provided the objectives in simplistic form. He reviewd the code as it was developed during the course of this project. His inputs helped us to make the code more secure and efficient. He provided us with different resources to look for help and overcome the challenges faced during the course.

REFERENCES

[1] Web Page. [Online]. Available: https://github.com/cloudmesh/client
[2] Web Page. [Online]. Available: https://github.com/cloudmesh/cloudmesh.cmd5
[3] Web Page. [Online]. Available: http://libcloud.readthedocs.io/en/latest/compute/drivers/ec2.html
[4] Web Page. [Online]. Available: https://aws.amazon.com/ec2/details/
[5] Web Page. [Online]. Available: https://pypi.python.org/pypi/requests
[6] Web Page. [Online]. Available: https://github.com/cloudmesh/cloudmesh.rest
[7] Web Page. [Online]. Available: http://docs.aws.amazon.com/AmazonSimpleDB/latest/DeveloperGuide/AboutAWSAccounts.html
[8] Web Page. [Online]. Available: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.htmll
[9] Web Page. [Online]. Available: https://github.com/cloudmesh/cloudmesh.aws

AUTHOR BIOGRAPHIES

Milind Suryawanshi received his BE (Electronics and Telecommunication) in 2010 from The University of Pune. His research interests include Big Data analytics for intelligence and research. **Piyush Rai** received his BE (Computer) in 2011 from The University of Pune. His research interests also include Big Data analytics for military intelligence and frinancial markets.

WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Milind Suryawanshi. Explored the libcloud EC2 drivers, investigated the various arguments requried by their different routines and implemented their use accordingly.

Piyush Rai. Worked on configurable parameters and database connectivity using pymongo and rest services.