Cloud Gateway

Rahul Bahal[1], Qianli Ma[1], Ayush Singh[1], and Mania Abdi[1]

[1]Northeastern University, Boston, 02115, USA
[1]{bahal.r, ma.qia, singh.ay, abdi.ma}@husky.neu.edu
[+]these authors contributed equally to this work

Description

The cloud gateway is a service to provide the ability to add more on-premises resources on demand without worrying about the scalability and resemble virtual tape libraries (VTLs). The problem arises when private clouds which have limited number of resources, temporarily, require more computational or storage resources. Users of public clouds would not face such problem because they are charged for the number of resources they use for a specific period of time. Thus, the hybrid cloud model may be employed as a cost-effective method of scaling resources to peak demand. The average-sized workload stays in the private cloud, and peak workload is provided in the public cloud. In this proposal, we propose Cloud Gateway (CG) as solution to this problem.

Cloud gateway, figure 1, helps the administrator of an OpenStack private cloud in providing seamless and secure integration between an organization's on-premises IT environment and cloud service providers. It is a virtual machine that runs on an OpenStack (or any) network to expand the local resource via a series of virtual private clouds. Machines will utilize the CG to send traffic between the private and public cloud subnets. Cloud software developers or users should not consider any changes to the environment they are working in. Amazon EC2 Driver is a commercial example of cloud gateways.

In this proposal we present a solution to implement CGs. In section 2, we identify the visions and scope of our work. Section 3 is dedicated to the proposed solution for implementing the cloud gateway. In section 4, the minimum acceptance area and features of this project. And finally in section 5, we provide our estimation about the time frame and release versions.

Example

Let's say originally we have 10 VMs working in a private cloud which use subnet IP address from 192.168.1.1 to 192.168.1.10. At peak demand, the admin wants to rent 10 more VMs in the public cloud and assign them with non-overlapping IP addresses from 192.168.1.11 to 192.168.1.20.

When the VM with the IP address 192.168.1.1 tries to communicate with 192.168.1.11, it's packet wont be able to reach the destination. Using the cloud gateways, a connection is established between different clouds and therefore the packets can be transferred between them. At the destination, the original packet can be retrieved by decapsulating the received packet.

Vision

Users

Any organizations who have a possible need to connect multiple different clouds (can be all private, public or the combination of these two kinds of clouds) and form a hybrid cloud will be our ideal user.

Use Cases

- You are the administrator of an OpenStack private cloud, and your goal is to allow your application team to seamlessly scale their applications for peak demand via a burst to the public cloud. A Cloud Gateway (CG) virtual machine runs on an OpenStack network to expand the local resource via an AWS Virtual Private Cloud (VPC). Machines will utilize the CG to send traffic between the private and public cloud subnets.
- As an administrator, your goal is to allow your application team to seamlessly connect different data centers owned by your company.

Scope

In this project we only focus of the possible solution of connecting subnets from multiple clouds. Our stretch scope would include fixing unwanted repercussions due to our code in the OpenStack system, e.g. memory leaks, security concerns, etc.

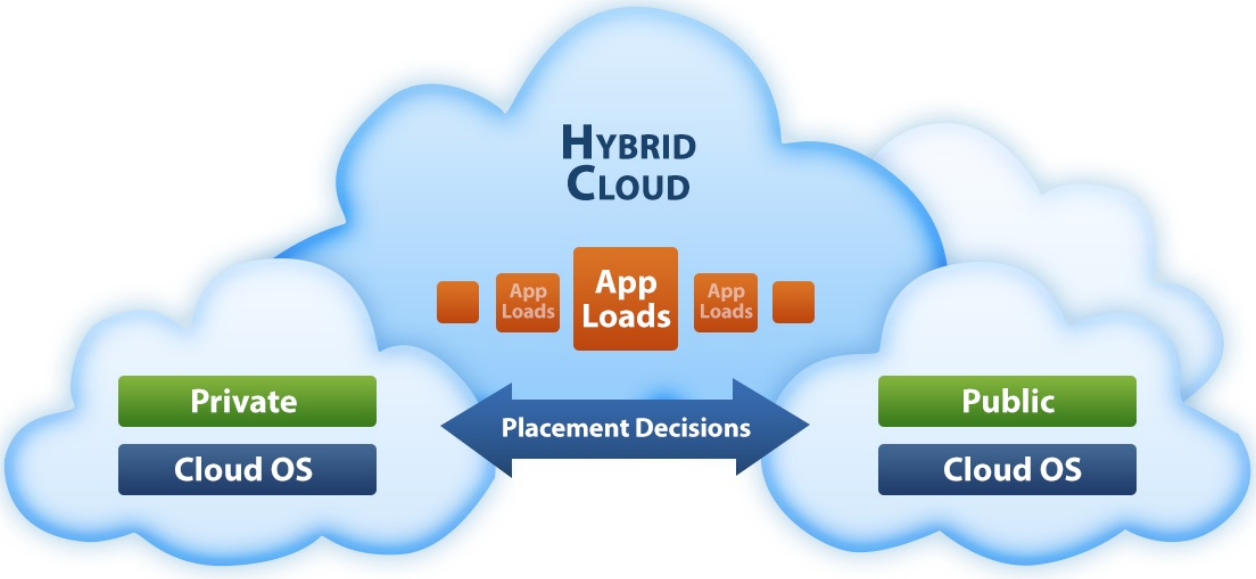

Figure 1. A Cloud Gateway (CG) system

Features

Topical subheadings are allowed.

- Cloud gateway should be able to
  - listen the traffic of its network.
  - identify any Internet connection required and use NAPT(network address and port translation) to enable and maintain such connection.
  - identify any needs of connecting different cloud and enable such connection by communicating with gateway of destination cloud.
- Configuration tool
  - An interface, graphical or command line, which enables users to maintain connections of multiple interconnected clouds, i.e. view, update, delete current connections and add new connections.

Goals

Developing a Cloud Gateway (CG) virtual machine that runs on an OpenStack network to expand the local resource and provision on premise compute-resources for seamless scaling of applications for peak workloads via public cloud, e.g. an AWS Virtual Private Cloud (VPC).

Solution

- Global Architectural Structure Of the Project The communication between the two subnets can be done with the help of a service, which will be similar to a Mapping Service that can handle the L2 and L3 communications. If a machine in a subnet wants to communicate to a machine in another network, it will first communicate with this service. In return, this service would then respond to the requesting entity with the address of the destination and the source would be able to send a message to the destination. Now, when the message is received by the destination, it will also communicate to the Mapping Service to confirm if the source is a valid one or not. If all checks out, the connection would then be established between the two entities.

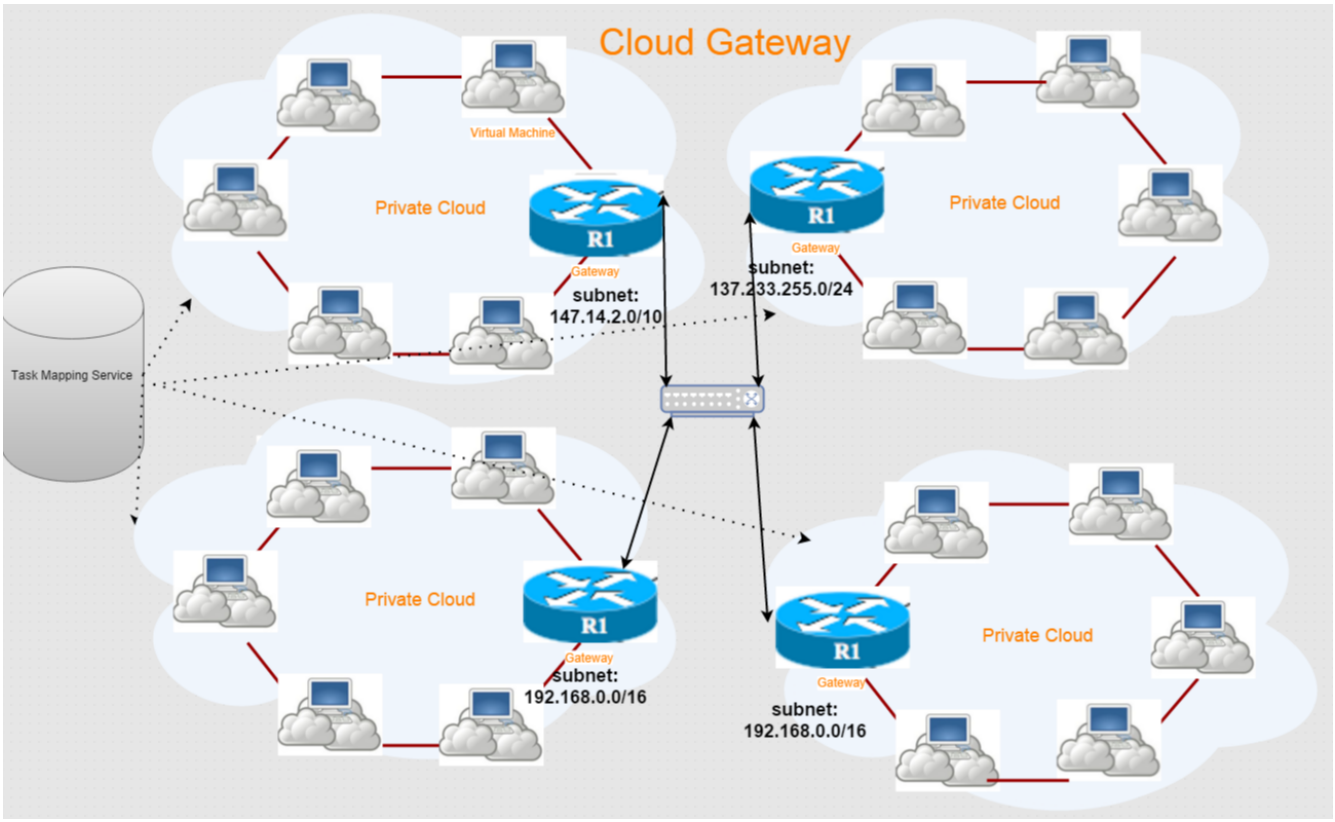

Figure 2. Proposed solution

- Design Implications and Discussion
  - Languages: Python, C, Bash.
  - Amazon AWS EC2, VPC
  - Amazon AWS APIs
  - Netstat, Socat, iptables, ntop, nmap, dig, IPplan
  - Wireshark, TCPdump, Zenmap

Acceptance Area

Minimum acceptance criteria will be to connect two subnets within the same network using the cloud gateway. In future, if time permits, we can look at achieving the following goals:

- Establish a connection between a subnet and an external network
- Integrate a Visual Monitoring UI
- Having a central framework that can drive the connections based on the resource requirements.

Release Planning

Release #1

- Study the AWS API, and other technology and theories to be used. Creating two subnets on different VPC for working environment.

- Successfully create connection using software between two VMs in two different cloud. This will simulate the connection of gateway. And this software will work as start point as a gateway program.

Release #2

- Connect subnet machine to VM(Gateway should be able to handle all outward packages)

Release #3

- Successfully create connection between two VMs from different clouds through cloud gateway through fixed code or manual control. At this point we should nail down the bridging mechanism of gateway.

Release #4

- Connect multiple VMs, i.e., connect subnets, in different VPC through gateway . This will verify the mechanism or connection management.
- Enable VMs in subnet to connect with internet As a further outcome of release 3
- After release 4 the core part of gateway should be considered finished and they should work correctly once set up.

Release #5

- Start working the configuration part of this gateway project. This should provide a easy way for managing connections among cloud, either through command line, configuration file or GUI. At release 5 we should have full design and at least implement some of the functionalities of control panel.
- Only with this part finished this project can be used in real world production environment.

Release #6

- Video giving the demo of the project
- Complete control panel and main development progress is finished.
- Fix minor bugs and make small improvement based on review.
- Improve API and documentation.
- Complete test platform and all major tests should pass.