1 Multi-cloud -- software install

In this lab you will get a taste for how you can create a single interface that supports a single query command-set to communicate with two very different cloud instances.

You will communicate requests to Amazon Web Services (AWS) and to Cloudstack. Amazon is a far more important player in this sphere, so the interface will use AWS command-set.

You will see that the configuration is specifically referred to as "ec2". Ec2 is the "compute" part of the AWS suite (it stands for Elastic Compute Cloud).

# Create a multi-cloud manager vm:

On Google Cloud Platform

Create an instance with these specifics:

name: multicloud1

region: us-west1

series: E2 - e2-micro (2vCPU, 1 GB memory)

boot disk: Ubuntu / Ubuntu 20.04 LTS (x86/64)

firewall: Allow HTTP & HTTPS

We will load cloudstack code in this system that enables Cloudstack to respond intelligently to aws ec2 queries.

We will use code in a docker container. The components used for this Cloudstack tool are very particular about the versions of dependent code. Happily, the container keeps all the right versions of components in sync.

# Configure for Docker:

On your multicloud1 system, logged in as root do…

Run the command --

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

A good return is "OK"

Add a new repository with the command -- (all on one line)

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

Run commands to make the new repository part of your search --

apt update  
apt-cache policy docker-ce

Install docker software --

apt install -y docker-ce

# Verify you have a good docker:

Check the status of your docker service --

systemctl status docker

It should show active (running) in green.

Run a docker command to be sure you docker install got important parts:

docker --version

This command should return a version and build number

# Install the docker container

Start a second ssh session to your multicloud1 system. We will need it in a moment to capture an ID from a docker session.

Since your docker service is operational, you can now install the cloudstack command management software. (note that ec2stack is the Amazon interface name)

docker pull cloudstack/cloudstack-ec2stack

Run the cloudstack multi-cloud software with --

docker run -ti cloudstack/cloudstack-ec2stack ec2stack-configure  
(do not proceed to answering the prompts yet)

In the **other ssh session** you started, issue the command --

docker container ls

Make note of the string under CONTAINER ID.

Back in your **first SSH session** (with the docker run command running) you will fill out the configuration of the tool… These are the values to supply:

Bind address 0.0.0.0

Bind port 5000

Cloudstack host: <outernetwork1 internal IP address>

Cloudstack port: 8080

Cloudstack protocol: http

Cloudstack path: /client/api

Cloudstack custom disk offering name: Custom

Cloudstack default zone name: zone1

Instance type mappings: no

Tag support: no

Still on multicloud1, issue the docker command -- (you will need the container ID you grabbed in your other ssh session)

docker commit <container id> ec2stack:mycloud

The "mycloud" is an arbitrary string identifier. A good result is a long hexadecimal identifier that starts with sha256.

# Run the configured docker container

Still logged into multicloud1 as root user do the following

Start up your docker container using the command --

docker run -d -p 5000:5000 ec2stack:mycloud ec2stack

This will set up a listener on port 5000 waiting for commands to send to a Cloudstack instance and an AWS instance.

# Install Standard AWS commands interface

Your docker instance will communicate with AWS using AWS cli (command line interface) style commands. AWS makes these available in a package.

apt install awscli