Brume Documentation

Release 0.1

Kushal Das, Praveen Kumar, Ratnadeep Debnath, Sayan Chowdhu

September 10, 2016

Contents

Contents:

CHAPTER 1

What is Brume?

Brume is a private cloud service which can be easily installed and maintained on standard user laptop/desktop or even on a server.

1.1 What is Cloud again?

We follow this definition of cloud. Brume is a very lightweight and minimally-featured Infrastructure as a Service (IaaS) system.

1.2 Why another cloud?

The existing FOSS IaaS systems target Enterprise as the primary user. This leaves a big hole where small/medium scale companies, universities, students can not install, or maintain a cloud of their own. The amount of technical knowledge required to maintain any of the available cloud is still very high. Brume will fit in this hole where it will give the minimal features of any existing cloud service.

If you install Brume, it will not destroy your existing network settings. One of the biggest setup point for starting is having a bridge device. In Brume, you will not require any such setup. It will work with the default Fedora/CentOS installation.

1.3 Do you want to become Openstack?

No. OpenStack is a big and complex project which can do all of things at the same time. It has lots of plugins, and can make use of different kinds of hardware for networking and storage. Brume will not have such capability. It will work on the standard libvirt on a Linux system. We are not trying to compete with any such complete IaaS systems.

1.4 Why not just use Vagrant then?

Vagrant is a great tool for setting up development environment using vm(s) on a system, but it is not a cloud. If you look back at the essential characteristics from the above mentioned document, you will find the following

- On-demand self-service
- Broad network access
- Resource pooling

- Rapid elasticity
- Measured service

These characteristics are not available in the existing tools as we know.

1.5 So will you be API compatible with Openstack or AWS?

Again, the answer is no :) That will require a lot of work, and might make the system unnecessarily complex. We don't want that.

1.6 Then how is this going to help a student/developer or even a SMC?

People will be able to use this to get cloud instances using the same cloud images which run on Openstack. People can write/test their configuration management scripts using this. For development teams inside a SMC, or an university, they can create/maintain cloud instances as required without going through IT, and IT teams (generally one person) will have much easier life while maintaining the actual infrastructure behind this.

1.7 Then tell us about the things you will provide in Brume

1. Cloud instances (the basic VM(s) you can create on a cloud).
2. Assigning elastic IP to the VM to get access from network.
3. Command line tools to manage the instances.
4. Web interface to manage the instances.
5. User/Groups with quotas.
6. A very simple object store.

The above list may change in future, but this is the initial goal we have.

CHAPTER 2

Development of Brume

Note: This is being actively developed. So you will find many things are getting changed regularly.

2.1 Dependencies

1. libvirt
2. kvm
3. iptables
4. redis
5. cryptography

2.2 Python 3.5

We are using Python 3.5 for the development work.

2.3 Typing hinting using mypy

We have type hinting enabled using mypy. If you are adding any new Python file, remember to add it to the *runmypy.sh*.

To setup mypy follow these commands below.

```
$ pyvenv env
$ source env/bin/activate
$ pip install mypy-lang
$ deactivate
```

CHAPTER 3

API documentation

This document contains the API documentation for brume. This is mostly for the communication between client and the server.

3.1 Authentication mechanism

The server expects the following two headers in every API call.

- x-username
- x-signature

3.2 Client <—> Server communication

This happens over the API calls, using the JSON data.

3.2.1 Uploading ssh keys

```
data = {'command': 'upload_ssh',
        'keyname': 'keyname',
        'content': 'ssh key file content'}
```

We will require a keyname (must be unique to the server), and the content of the ssh key file.

Output:

```
{'ouput':res, 'message':msg}
```

res is a boolean values, the message can give more details, including an error 500 one.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search