dietstack Documentation

Release 0

dietstack

Jun 10, 2018

Contents

1 User's Guide **3**
- 1.1 Foreword . . . 3
- 1.2 Instalation . . . 4
- 1.3 User guide . . . 7
- 1.4 Development . . . 9

Dietstack is minimalistic OpenStack distribution which is focused on simple deployment, transparent management and easy upgrades.

CHAPTER 1

User's Guide

1.1 Foreword

Whole project started as a hobby project for fun. I always wanted to have small, cheap private cloud at home. I have built couple of OpenStack private cloud solutions so I decided to use the experience and prepare OpenStack distribution which is as easy to install and manage.

All services runs in Docker containers and basic deployment is done by simple bash script. There is no need to hassle with config files, because these are generated by DietStack containers during startup process. Simple deployment means that unlike other OpenStack distributions, DietStack supports exactly one configuration based only on free linux software. It means that it uses KVM (libvirt) as a hypervisor for Nova, NFS for Cinder, VXLANS and linux bridges for Neutron networking.

To avoid dependencies as much as possible, it doesn't use packages from any linux distribution to deploy OpenStack service. Instead docker containers are built directly from upstream OpenStack repositories. This makes DietStack independent from major linux distros. DietStack doesn't use any configuration management for deployment. Is is just few bash scripts.

HA configuration is not supported as it increases complexity significantly. Availability of small installations can be solved by other ways.

1.1.1 OpenStack services in DietStack

- Keystone (Identity Management)
- Glance (Images)
- Nova (Compute)
- Neutron (Networking)
- Cinder (Block Storage)
- Heat (Orchestration)

1.1.2 OpenStack version

Version 1 of DietStack was based on `Newton` version of OpenStack. Current version is version 2, and is based on `Pike`.

1.1.3 How DietStack can be used

Diestack is suitable for OpenStack training and learning purposes as well as for small private cloud installations. At home I'm using it for all my VMs like Git server, NAS Server, Torrent Downloader and more. I'm using it also for development purposes and due to the fact OpenStack has built-in multitenancy, I can have production and development VMs separated on a network level.

1.1.4 Scalability

DietStack is built to run OpenStack on cluster up to 10 nodes - 1 control node and 9 compute nodes. It should be enough for most SOHO purposes. Nevertheless technically there is no built-in limit in DietStack itself, so you can try to deploy it on more nodes if you want.

Continue to *Instalation*

1.2 Instalation

1.2.1 Architecture

Unlike other OpenStack distributions DietStack supports only one configuration. There is a Control-Node, where Mysql, Rabbitmq and all OpenStack server components are running. In most basic installation, you can deploy everything on one node so Control-Node is also Compute-Node. This way you can run DietStack locally on your notebook or desktop for development, testing or training purposes.

1.2.2 Networking in DietStack projects

Each instance is connected to one or more private networks which are not directly accessible from outside of a cloud. Instances that are not on same private network cannot communicate together without router. It is very similar to the way how pysical networking is working in real world. There is a concept of Floating IP, which are public routable IPs and these can be optionally assigned to private IPs of VM and have access to VM from outside of a cloud. DietStack does not implement Ditributed Virtual Router (DVR) technology which makes networking more robust, but makes physical networking more complex.

1.2.3 How to connect DietStack to your physical network

As written in previous topic, Floating IPs needs to be externaly routable in order to users are able to connect to instances created by DietStack. It means that you need to choose physical network, where you need to have a pool of IP addresses which are dedicated for that purpose - will be assigne by Neutron and not by your DHCP server). Floating IP rabge can be then specified during creation of external network during DietStack initial configuration (see `first_vm.sh` script in container invoked by `dscli.sh`)

1.2.4 1. Local Installation

All containers will run on your local computer.

Requirements

1. At least 8GB RAM
2. At least 10GB of free space
3. Debian 9 or Ubuntu 16.04 on your computer
4. `root` privileges or ability to get then with `sudo`

```
git clone https://github.com/dietstack/dietstack.git --recursive
cd dietstack
sudo ./ds.sh
```

Now you can connect to Horizon Dashboard :) - http://192.168.99.1:8082

You can also run OpenStack client to manage OpenStack cloud from cli:

```
./dscli.sh
```

1.2.5 2. DietStack in Vagrant

DietStack runs in two and more VMs conrolled by Vagrant.

Requirements

1. At least 8GB RAM
2. At least 10GB of free space
3. kvm enabled and Vagrant with `vagrant-libvirt` plugin installed

Then you can run:

```
git clone https://github.com/dietstack/dietstack.git --recursive
cd dietstack/vagrant
vagrant up
```

You can connect to Horizon Dasboard on http://192.168.99.2:8082

You can also run OpenStack client to manage OpenStack cloud from cli:

```
vagrant ssh control.dietstack
sudo su -
cd /root/dietstack
./dscli.sh
```

1.2.6 3. Multi-Node Installation

This is installation you will need to use when you would like to use DietStack for the real purposes.

Requirements

1. Debian 9, Ubuntu 16.04 or Ubuntu 18.04 already installed on all nodes
2. At least 8GB RAM on Control-Node
3. At least two ethernet interfaces on Control-Node (one for external connectivity and one for DS network)
4. At least one ethernet interface on each Compute-Node (DS Network)
5. One switch (1G/10G) to connect all nodes (DS Network)
6. Couple of ethernet cables to connect nodes with a switch
7. Dedicated block device (mirrored if possible) for Cinder volumes (optional)
8. Dedicated block device for Mysql backups (optional)
9. `root` privileges or ability to get then with `sudo` on all nodes

Network diagram

Control node

First and most complex step is to prepare networking, interface for external connectivity and interface for DS network.

For external connectivity you need setup `br-ex` interface. It is a `linux bridge` and if we want to have connectivity for the endusers, we need to add physical interface into it. Let's say interface *ens3* is connected to switch to your network and will be used by users to access VMs. Your external network has subnet 10.0.0.0/16, IP address on ens3 interface is 10.0.0.2 and default gw is 10.0.0.1.

Create file called `/etc/network/interfaces.d/br-ex` with following content:

```
auto ens3
iface ens3 inet manual

# Bridge setup
auto br-ex
iface br-ex inet static
        bridge_ports ens3
        address 10.0.0.2
        netmask 255.255.0.0
```

Remove `ens3` lines from `/etc/network/interfaces` and install `bridge-utils` package:

```
apt-get install -y bridge-utils
```

Now you can setup your DS network interface. DS network is used for communication between openstack services on all nodes, for vxlans tunnels and for nfs mounts, and it has to be separated from external network. Do not use DHCP in DS network, but use static assignment. Let's say name of your DS interface is `ens4`. You can freely choose same subnet as we did (192.168.1.0/24), so `192.168.1.1` for you control node is OK.

Create file `/etc/network/interfaces.d/ds-net` with following content:

```
auto ens4
iface ens4 inet static
    address 192.168.1.1
    netmask 255.255.255.0
```

Now you can reboot your node. After the reboot, ensure that your `br-ex` interface is up and have ip address assigned (`ip a s`). Do the same for your DS network interface.

If everything is correct, you can install Dietstack on control node:

```
sudo su -
git clone https://github.com/dietstack/dietstack.git --recursive
cd dietstack
EXTERNAL_IP='10.0.0.2/16' DS_INTERFACE=ens4 ./ds.sh
```

- DS_INTERFACE=ens4 - `ens4` is interface physicaly connected to DS switch
- EXTERNAL_IP='10.0.0.2/16' - needs to be set in order to horizon spice console works (same as on control node)

Block storage for Cinder service

Control node is also storage node in DietStack installation. It means that when you create volume in Horizon interface writes will go over NFS to control node and to block device we will configure here. Block device needs to be mounted to `/srv/dietstack/cindervols`.

Compute nodes

Installation of compute node is much easier than installation of control node. You just need to clone the repository and run the `ds.sh` with correct parameters:

```
sudo su -
git clone https://github.com/dietstack/dietstack.git --recursive
cd dietstack

COMPUTE_NODE=true DS_INTERFACE=ens4 CONTROL_NODE_DS_IP=192.168.1.1 ./ds.sh
```

So you basically need to set 4 variables:

- COMPUTE_NODE=true - we are installing compute node
- DS_INTERFACE=ens4 - `ens4` is interface physicaly connected to DS switch
- CONTROL_NODE_DS_IP=192.168.1.1 - tells to DietStack where to find control node in DS network

Continue to *User guide*

1.3 User guide

1.3.1 Post installation notes

After installation of Control node, you can will have empty private cloud running. You have no external network defined, no flavors, no images, no volumes, no ssh keys. So you cannot run any instance at the moment. In this stage you have a chance to learn how to get Openstack cloud into usable state. You can do that in Horizon interface or in

CLI utilizing Openstack client, but DietStack has prepared bash script `first-vm.sh` in administration container called `osadmin`.

1.3.2 Settings file

On each node, where you run `ds.sh` you will find file `/srv/dietstack/settings.sh` where DietStack stores all important variables. DietStack will use that file for all subsequent executions.

If you would like to change settings of existing installation you can change variables in `settings.sh` and rerun `ds.sh`.

1.3.3 Administration container

In order to manage your DietStack installation with Openstack client there is a container called `osadmin`. To get to DietStack cli, just run ./dscli.sh script on Control Node and you will get to container where you can find many useful utilities. One of most important is `Openstack client`.

If you want to manage OpenStack cloud, you need to authenticate against it. In osadmin container you can find two rc files:

- `adminrc` -> will make you administrator of a cloud
- `demorc` -> demo user in demo project

For instance, if you would like to list all users you have to be admin user, and use openstack client to see all the users:

```
$ . adminrc
$ openstack user list
```

1.3.4 First Instance

Once you are in the administration container, you can prepare the cloud for running your first instance with helper script:

```
$ ./first_vm.sh
```

If script finishes succesfully there are several things configured:

- Created external network called `external_network`
- Created m1.nano flavor (64MB RAM, 1GB disk)
- Created `internal_network` in `demo` project (192.168.35.0/24)
- Keypair mykey is added into `demo` project
- Instance `first-vm` is started
- Floating IP to `first-vm` is assigned

1.3.5 Log files

Each DietStack docker container logs internally into container and log directory from container is accessible on host in directory `/srv/dietstack/logs/`

1.3.6 Stopping DietStack

Run as root from dietstack directory:

```
sudo ./utils/destroy-ds.sh
```

Warning: This will destroy all data in database on control node

1.4 Development

Whole development is done on GitHub DietStack organization.

Each openstack service has its own git repository (docker-service_name) and is automatically built and tested on Buildbot server (not publicly available yet). If test of container succesfuly passes, container is pushed to DockerHub

We are working on bigger integration nightly testing with a tempest openstack project.

1.4.1 Versioning

TODO

1.4.2 Contribution

Simply fork the repostiroy

DietStack consist of several git repositories. The main repository is dietstack/dietstack where main deployment script `ds.sh` is located and then separate repositories for each openstack service. If you want to contribute to any repository just simply *fork it <https://help.github.com/articles/fork-a-repo/>_*, make changes, run the `test.sh` and if test passes, create a pull request.

In order to test your changes to any project you need to have main dietstack repository cloned. Then change the versions

1.4.3 Development environment

In order to contribute to development of DietStack you need to have Linux running on your machine. I'm using Ubuntu, but I'm sure that any ditribution where Docker runs can be used for development. Minimal configuration for development is at least 16 GB of ram. Then you need to install following packages:

- Docker
- Vagrant with `vagrant-libvirt` plugin (sudo apt install vagrant; vagrant plugin install vagrant-libvirt)
- libvirt-bin and qemu-kvm for virtualization in vagrant

1.4.4 Testing

TODO