Hydra Documentation

ownCloud

Rasmus N, Leo B, Philip B, Shenée B, Sebastian H

October 13, 2021

Abstract

Documentation for OUMPH cloudification
Write more smart stuff here!

Contents

1 Networks **2**
1.0.1 DMZ . . . 2
1.0.2 Film/foto . . . 2
1.0.3 HR/ekonomi . . . 2
1.0.4 R&D . . . 2
1.0.5 IT-admin . . . 2
1.0.6 SAN 1 . . . 2
1.0.7 SAN 2 . . . 2
1.1 DNS . . . 2
1.1.1 Google . . . 2
1.2 Ip Ranges . . . 2

2 Security groups **3**
2.1 Web . . . 3
2.2 SSH . . . 3
2.3 AnsibleMaster . . . 3
2.4 DatabaseGroup . . . 3
2.5 FileserverGroup . . . 3

3 Instances **3**
3.1 Default network settings . . . 3
3.2 Web . . . 3
3.3 NAS . . . 3
3.4 AnsibleMaster . . . 3
3.5 VmServer . . . 4
3.6 DbServer1 . . . 4
3.7 FileServer1 . . . 4
3.8 DbServer2 . . . 4
3.9 FileServer2 . . . 4

4 Docker Services **4**
4.1 NextCloud . . . 4
4.1.1 Config . . . 4
4.2 Jitsi . . . 4
4.3 Matrix-synapse . . . 4

5 Other Services **4**

6 **Major changes from order** **5**
6.1 Network . . . 5
6.1.1 Subnet sizes . . . 5
6.1.2 Ip-ranges . . . 5
6.2 Instances . . . 5
6.2.1 VmServer(2) . . . 5
6.2.2 AnsibleMaster . . . 5

7 **Terraform** **5**
7.1 Prerequisites . . . 5
7.2 Usage . . . 5
7.2.1 First time . . . 5
7.2.2 Build . . . 6
7.2.3 Destroy . . . 6
7.3 Future features . . . 6
7.3.1 Volumes . . . 6
7.3.2 VPN . . . 6

1 Networks

1.0.1 DMZ

192.168.1.0/24

1.0.2 Film/foto

192.168.2.0/24

1.0.3 HR/ekonomi

192.168.3.0/24

1.0.4 R&D

192.168.4.0/24

1.0.5 IT-admin

192.168.5.0/24

1.0.6 SAN 1

192.168.6.0/24

1.0.7 SAN 2

192.168.7.0/24

1.1 DNS

1.1.1 Google

¿ 8.8.8.8 alt 8.8.4.4

1.2 Ip Ranges

Network ID	Network Interfaces	Server Range	Host ID Range
192.168.1.0	192.168.1.1 - 192.168.1.49	192.168.1.50 - 192.168.1.99	192.168.1.100 - 192.168.1.254
192.168.2.0	192.168.2.1 - 192.168.2.49	192.168.2.50 - 192.168.2.99	192.168.2.100 - 192.168.2.254
192.168.3.0	192.168.3.1 - 192.168.3.49	192.168.3.50 - 192.168.3.99	192.168.3.100 - 192.168.3.254
192.168.4.0	192.168.4.1 - 192.168.4.49	192.168.4.50 - 192.168.4.99	192.168.4.100 - 192.168.4.254
192.168.5.0	192.168.5.1 - 192.168.5.49	192.168.5.50 - 192.168.5.99	192.168.5.100 - 192.168.5.254
192.168.6.0	192.168.6.1 - 102.168.6.49	192.168.6.50 - 192.168.6.99	192.168.6.100 - 192.168.6.254
192.168.7.0	192.168.7.1 - 192.168.7.49	192.168.7.50 - 192.168.7.99	192.168.7.100 - 192.168.7.254

2 Security groups

2.1 Web

Allow port 80 from any ip Allow port 443 from any ip Allow port 22 from 192.168.5.51 Deny otherwise

2.2 SSH

Allow port 22 from 192.168.0.0/16

2.3 AnsibleMaster

Allow port 22 from 91.229.245.181

2.4 DatabaseGroup

Allow port 3306 from 192.168.0.0/16

2.5 FileserverGroup

Allow port 3306 from 192.168.0.0/16

3 Instances

3.1 Default network settings

whitelist:
lo: INPUT and OUTPUT
90.229.245.181:22 INPUT and OUTPUT
LAN:22 INPUT and OUTPUT
RELATED,ESTABLISHED INPUT and OUTPUT
LAN:80,443 INPUT and OUTPUT
LAN:8.8.8.8,8.8.8.4.4 INPUT and OUTPUT
default policys:
INPUT, OUTPUT and FORWARD DROP

3.2 Web

ip: 192.168.1.50
open ports:
WAN: 80, 443
LAN: 22

3.3 NAS

ip: 192.168.5.50
open ports:
LAN: 22 INPUT
(Only open to inside network 192.168.0.0/16)
closed ports:
LAN: INPUT DROPPED

3.4 AnsibleMaster

ip: 192.168.5.51 (floating ip: known after build)
open ports:
WAN: 22
(only to specific ip)

3.5 VmServer

ip: 192.168.6.50
open ports:
LAN: 3306, 8080, 8096, 9980 INPUT and OUTPUT
(Only open to inside network 192.168.0.0/16)

3.6 DbServer1

ip: 192.168.6.51
open ports:
LAN: 3306 INPUT and OUTPUT
(Only open to inside network 192.168.0.0/16)

3.7 FileServer1

ip: 192.168.6.51
open ports:
LAN: 445 INPUT and OUTPUT
(Only open to inside network 192.168.0.0/16)

3.8 DbServer2

ip: 192.168.7.50
open ports:
LAN: 3306 INPUT and OUTPUT
(Only open to inside network 192.168.0.0/16)

3.9 FileServer2

ip: 192.168.7.51
open ports:
LAN: 445 INPUT and OUTPUT
(Only open to inside network 192.168.0.0/16)

4 Docker Services

4.1 NextCloud

Hostname: VmServer ip: 192.168.6.50 ¡?¿ port: ¡?¿

4.1.1 Config

docker/nextcloud/config ¡?¿

4.2 Jitsi

Hostname: VmServer ip: 192.168.6.50 port: 8089

4.3 Matrix-synapse

Hostname: VmServer ip: 192.168.6.50 port: ¡?¿

5 Other Services

Do we even have any?

6 Major changes from order

6.1 Network

6.1.1 Subnet sizes

¡spellcheck¿ Originally the subnets were set with /27 netmasks.
We felt this was unneccacarily complicated as it makes it much harder to troubleshoot.
To make troubleshooting easier we made /24 networks, one for each department/relevant section.

6.1.2 Ip-ranges

We also set some rules for what different ip addresses *should* mean.

192.168.x.1 - 192.168.x.49: network units
192.168.x.50 - 192.168.x.99: servers
192.168.x.100 - 192.168.x.254: clients

This setup makes it easy to identify device based on the ip address when inspecting logs during troubleshooting
The only downside with this is that every adress from 1-99 has to be set manually, and the dhcp sets the clients ipadresses.
If you want to know which department a request came from, or is going to, x in the example will tell you what to look up in the documentation.
In hindsight, having 49 adresses for network units is quite overkill on the cloud, with what is supposed to run on it as it looks today. The thought to have that many emerged from the real-world setup scenario, where the company wanted ip-cameras, which then would be put in the 1-49 range.
This could have also been done by having a dedicated subnet for the ipcameras, but it feels nice to keep general rules to make it quick and easy to get an idea of what's happening reading logs.

6.2 Instances

6.2.1 VmServer(2)

Since we are running on the cloud, and after some discussion with our consult *hxp*, we decided it was unneccecary to have two vm-servers.
The server in question was supposed to be used as a backup incase the main vm-server goes down. But since it's only supposed to host 'in-house' applications, the risk of getting overloaded is small enough that the cost is too great compared to what we would gain.
If the server goes down, it will be easy to get it up and running again, and since we are using volumes, the data is persistant. We would get some downtime, but no data loss.

6.2.2 AnsibleMaster

We decided to use ansible to configure all the machines. To make the process as easy as possible, we wanted to have the ansible master running inside of the cloud. This also means that we don't need to open the ssh-ports to wan on every instance.

7 Terraform

7.1 Prerequisites

This script assumes you already have a router with a specific ID in openstack
Change the ID in `terraform/variables.tf` to match yours.
It also assumes that you have a ssh key-pair on openstack named `Nietzsche`,
aswell as the private key named `id_rsa` in `ansible/.ssh/id_rsa` You will also need to replace the set ip in the security group allowing ssh from WAN.
In order to give your credentials to openstack
you will need to source the project-script from openstack before running terraform commands.

7.2 Usage

7.2.1 First time

```
terraform init
```

7.2.2 Build

`terraform apply -auto-approve`
For some reason, terraform needs to run the apply command twice to actually set the security groups.
This will make the file-transfer a bit weird.
Terraform is going to time-out on trying to ssh into the machine ”AnsibleMaster”,
you can cancel this with `^c` and re-run the the apply command.
This will set the security groups, and make it possible to transfer the files over ssh.

7.2.3 Destroy

`terraform destroy`

7.3 Future features

7.3.1 Volumes

We want to link volumes to some machines, to be able to have persistant data, and logs.
The idea is to take snapshots of the important data and our logs,
and have those snapshots on rolling backups.

7.3.2 VPN

We want to set up a VPN server/service so that a ’client’
can work inside the cloud as if you were connected in an office.