Linux Project Milestone 3

DNS-as-a-service

Team 10

Name	- Unity ID
Bhavya Dwivedi	- bdwived
Harsh Pathak	- hpathak
Khantil Choksi	- khchoksi
Shubhankar Reddy	- skatta2

Advisor / Mentor: Prof. Anand Singh

Course: CSC/ECE 792 Linux Networking
Fall 2018
North Carolina State University

Table of Contents

1. Introduction

1.a. Background

A Domain Name System (DNS) is essentially the phone book for any network – including the internet. Every time a user surfs the web, they use DNS. Without it, each user would have to remember IP addresses of the site s/he wanted to visit. For instance, instead of remembering a hostname like www.google.com, s/he would have to remember 8.8.8.8 and all the other load balanced IP addresses. DNS is a function responsible for mapping the hostnames to IP addresses for other applications on the Internet.

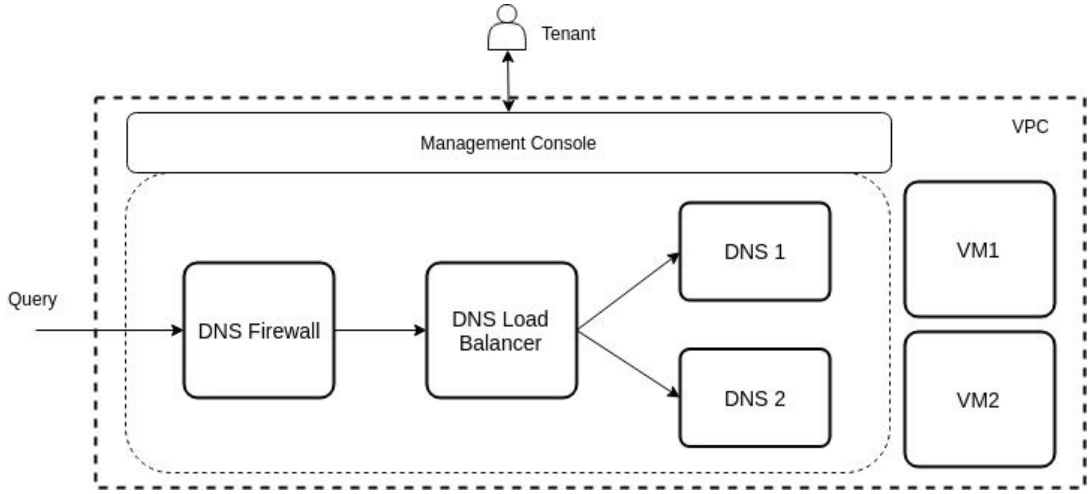
1.b. Problem Statement

DNS maintains a database of hostnames - IP address mapping and defines the protocol to exchange this information. It is designed to be a hierarchy of nameservers maintained in a distributed manner. It is a critical component of the Internet, which also makes it the most vulnerable to mismanagement and attacks. To quote Sir Tim Berners-Lee, “The Domain Name Server (DNS) is the Achilles heel of the Web. The important thing is that it's managed responsibly.”
Following are the challenges that are faced in managing the traditional DNS:

1. Configurational Errors
2. DNS Outages - No automated failover mechanism
3. Poor Performance – High DNS Latency, High TTL
4. Security Attacks – DDos, DNS Flood
5. Inefficient Traffic Routing
6. Scaling

1.c. Project Description

With the ever-increasing scale of the web and advent of the cloud computing, the cost of operations has been brought down and ease of management has increased significantly. Infrastructure-as-a-service has quite evidently attracted a lot of web enterprises. Similarly, DNS is also one of the services that have the benefits of moving to the cloud. However, some challenges in the management of DNS mentioned above still remain. To mitigate those challenges, automation has been introduced and management features are provided.
With DNS as a Service (DNSaaS), customization is offered in services for each user to a certain extent such as providing DNS request routing based on metrics such as latency, geolocation, failover, weighted round robin on the IP addresses serving a particular web page or application instance. Also, customer pays the provider for the number of queries resolved to its domain name rather than hosting its own DNS. With security ensured against attacks such as DDoS and DNS Flood, customer has an inherent commercial benefit the provider need not be paid for the malicious queries filtered at the firewall itself.
Basic functional diagram of DNS-as-a-service is shown below:



1.d. Summary

In this project, we aim to develop a DNS solution which eliminates the shortcomings of a traditional DNS by employing automation techniques and cloud technologies. We have developed our solution to to provide hassle-free DNS deployment and management services to the customer in both VM and Container modes.

2. Related Work

This section enumerates, in brief, the DNS features and compares the implementation by successful commercially available products in the market like Microsoft Azure DNS, Amazon Route53, Google Cloud DNS and Cloudflare.

- **Performance**: Azure DNS provides monitoring of the number of queries, DNS records and percentage utilization of DNS records of each DNS zone. Cloudflare DNS supports monitoring of the number of queries by type of response codes and DNS latency. NS1 supports monitoring of the number of queries by geography, server instances and type of traffic.

- **Traffic Management:** Azure provides traffic management with automatic failover based on traffic routing methods, the health of endpoints and also provides a feature of Nested traffic manager profiles for complex networks. Cloudflare does per data center failover and RR load balancing along with Geo-steering for traffic management. Route-53 provides several rules such as Geoproximity, Geolocation, priority, multivalue, latency, and failover for traffic management. Dyn DNS provides geolocation and ratio based traffic management with active failover.

- **Security:** Providers such as Cloudflare, Incapsula, and Dyn provide DDoS protection in their DNS services. They provide these along with high availability, DDoS mitigation, and faster connections by using anycast routing.

- **Management User Interface**: It is a web-based GUI which lets tenant manage DNS records without knowing about the inner functionality of the system. E.g. AWS Route 53 provides a console to create, update hosted zones, visualize the health checks, configure traffic policies and traffic records. Similarly, Azure provides Azure Portal and Google Cloud provides GCP Console having following features:
    a. Configuration Automation:
    - Using the REST API calls, a tenant can automate the process of creating DNS routes, giving weight, configure health checks, in their YAML or Python automation scripts.
    - E.g Ansible has a module called "route53" to automate the configuration for DNS routes.
    b. Rollback Automation:
    - If traffic is being diverted to a new server implementing a new feature, and the metrics for the new server is not as expected, then using automation scripts, the tenant can rollback to the original, steady server (i.e. by diverting back traffic to a steady server).

- **Private DNS**: Private DNS records let a user create private hosted zones and route traffic using easily managed domain names within user's VPCs. This can, for instance, allow the user to quickly switch between IP-based resources without the need to update multiple embedded links.
  - The advantage of using Private DNS is that, if a domain name is migrated to another server, there is no need to change any nameservers and the domain names will automatically point to the new location.

Features	AWS Route 53	Google Cloud DNS	Azure
Zone	Private Hosted Zone	Managed Zone	DNS Zone

Support for most DNS record types	Yes	Yes	Yes
Anycast based servicing	Yes	Yes	Yes
Domain registrar	Yes	Yes	No
Latency-based routing	Yes	No	No
Geography-based routing	Yes	No	Yes
DNSSEC	Not for DNS service, only for Domain Registration	Yes	No

3. Project Description

3.1. Objective

The project goal is to provide DNS-as-a-service in Virtual Private Clouds along with functional and management features as described in the next section. Our objective is to automate the process of VPC, subnet and DNS CRUD operations and configurations as per the input received from tenants, providing the tenants a seamless service.

3.2. Features

3.2.1. Functional Features

- **Load Balancing:**
    - The load being the requests received, the DNS balances it by distributing the incoming DNS requests to it from clients in a VPC.
    - The user can configure the load balancing mechanism as in whether one particular type of request should get resolved from a particular server (static) or if requests should be divided proportionally/equally among DNS servers (dynamic).

- **Security:**
    - The DNS servers make sure to block IP addresses when an unusually large number of requests originate from a single IP (The user can choose whether to block the IP permanently, for a specific span or limit the number of requests from it). By doing so, the DNS server prevents the possibility of a DoS attack on the customer's web servers/infrastructure.
    - Each of the DNS servers maintains an access control list, which it uses to establish whether an incoming request is valid or not. We execute a background script for the named.conf file verifying the current content of the file with the previously stored checksum, if someone hacks into one of the servers, to replace records with their own malicious website, the encryption and HMAC stored will prevent the root server from delivering the false information to the client, also help it identify the server which was compromised.

3.2.2. Management Features

- **User Interface / API:**
    - Web-based Graphical User Interface:
    - It provides each tenant to create, update and delete DNS records using web-interface, without configuring them inside their VPC using CLI.
        - REST API Endpoints (CLI commands but can be extended easily for REST API):
        - Using the REST API endpoints, tenants can build their custom automation and configuration scripts in Python or Bash.

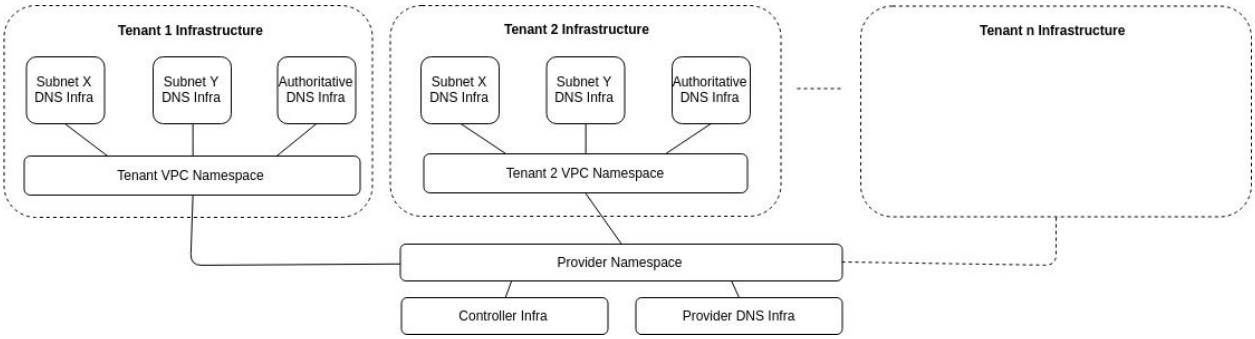
- Example, Using API call, a tenant can send POST API call to create DNS record entry with parameters like VPC id, domain name, weight, TTL, type. In response to that, our DNS-as-a-Service will take the necessary steps to create a new DNS record entry for that tenant in his specific VPC.

- **Resiliency and Availability:**:
    - The DNS server can provide high availability by maintaining an active-passive resiliency paradigm on the basis of a user-configurable threshold of errors count and health-check status of each instance and switch to the redundant instance once the threshold is crossed.

3.2.3. Kube Inspired Features:

- Kube DNS inter-services communication:
    - We have implemented the kube inspired feature of communicating between two services without having any additional configuration. So, let's say in a VPC, client want to run two different services, then these two services will be able to communicate each other by service name.
- Kube High Availability and Automatic Recovery of containers:
    - Inspired by Kube DNS, we have developed and integrated a feature to spawn a new container with the same configurations (maintaining state) whenever any container DNS is stopped in order to provide auto-recovery/ self-healing feature.

3.c. High Level Topology



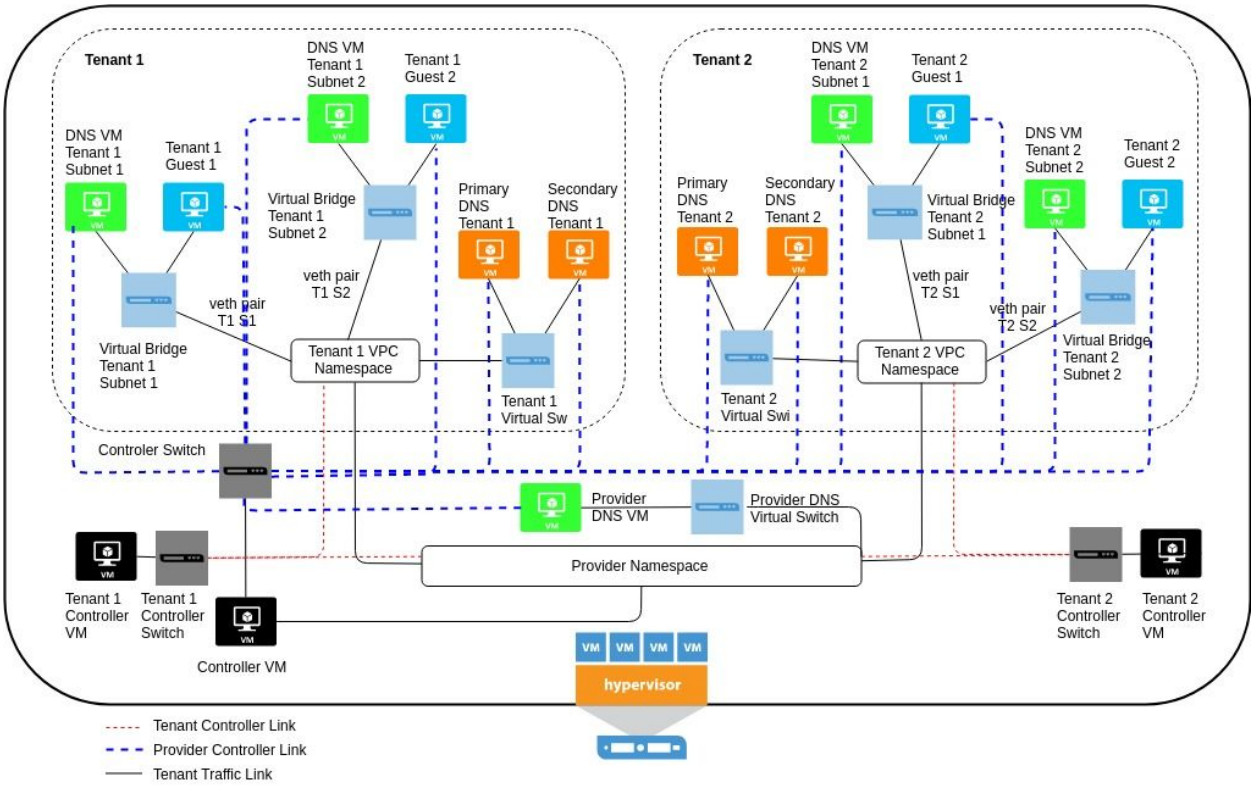
3.d. Environmental Constraints

- DNS configure automation is easily done in the case of docker containerized environment, as we can mount the volumes of containers to hypervisor. But, in case of VM instances as DNS servers, it is hard to automate the DNS configuration and services communication.
- In case of VMs, the time spent in creation of VM and the memory and space occupied by VM was much more . While , in case of containers, the whole process was very fast.rd to automate the DNS con

4. Implementation Architecture

4.1. VM Deployment

4.1.1. VM Implementation Architecture

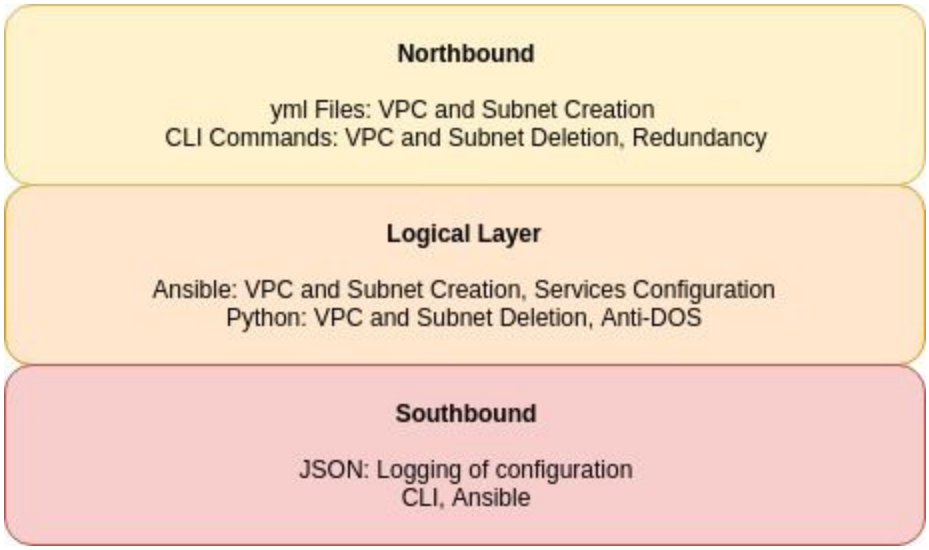


Role of each component:

1. Provider Infrastructure
    a. Provider namespace: provides connectivity to and from the Internet and masquerades all traffic from VPCs to the internet, also prevents communication across VPCs.
    b. Provider DNS VM: responsible for resolving external DNS, it is also the top level domain DNS server.
    c. Primary and Secondary DNS servers: per tenant high available pair which are nameservers and maintain DNS entries for DNS VM in each subnet.
    d. DNS VM Tenant 1 Subnet 1: per subnet DNS server in the VPC, which is the authoritative server for all application VMs in the subnet.
2. Controller Infrastructure
    a. Controller VM: Allows the provider to execute scripts to provision infrastructure for each tenant.

    b. Tenant 1 controller VM: per tenant VM to allow clients to log into and manage their VMs
3. Tenant Infrastructure
    a. Tenant1 Guest1: VM which hosts the application used by the tenant.
    b.
4. ok

4.1.2 Functional Diagram of VM Architecture



DNS Hierarchy Explanation:
North - South Traffic and logical layer design

1. The DNS server at the hypervisor is the top level domain (TLD, Level1) DNS server for the project's infrastructure. All north-south traffic's DNS queries are resolved by the TLD server. This server maintains records for each of the VPCs in the hypervisor.
2. Each tenant is allocated a unique namespace to maintain isolation among multiple tenants. Each of these namespace consists of multiple subnets, each served by a veth interface executing dnsmasq on one end, and a switch on the other. This ensures that any VMs created in the subnet are assigned IPs automatically. Each namespace also has 2 DNS servers executing in it, at the namespace level(Level2). Both of them execute the keepalived to ensure high availability in case of DNS VM failures(Level2). Every subnet is also allocated a DNS server, at the subnet level(Level3).
3. This Level2 DNS(Nameserver) server is responsible for maintaining the records for every authoritative Level3 DNS server and points to them.
4. All the DNS servers are connected to a controller switch and a controller VM which can provision new infrastructure for new tenants.
5. All queries for the north-south traffic end up at the TLD, Level1 DNS server. The DNS server looks up its forward zone records. If a zone record exists and can be resolved locally it returns the A record for the particular domain. This server also maintains views for geolocation based routing and redirects

traffic based on the incoming source IP address to the Level2 server which is responsible for serving content for that region. If a zone record exists and it cannot be resolved locally, it requests for the A record from the respective Level2 server which is pointed by the NS record in its forward file.

6. The Level2 server looks up its records and if it can be resolved locally, returns the A record for the particular zone, else returns a “record not found” message to the Level1 server. If the zone exists but cannot be resolved locally, it tells the Level1 server the IP of the particular Level3 which is responsible for that zone(content-based routing).
7. The Level1 DNS server on receiving the message queries the Level3 DNS server (Authoritative) for the IP of the URL. The Level3 DNS server responds to the query with the IP address of the server responsible for that domain (round robin or static).

East-West traffic

1. The implemented design only allows east-west traffic within the VPC and not across VPCs. Each VM in the hypervisor has its DNS server (in /etc/resolv.conf) configured to the Level2 DNS server’s IP and contacts it for all domain resolutions within the VPC.

4.1.3. User Guide

Steps to set-up DNS

1. Install bind9 packages on all DNS servers.

2. Configure DNS server by editing the /etc/named.conf file
   nano /etc/named.conf
   a. Add IP of the DNS VM in options as follows:
      listen-on port 53 { 127.0.0.1; <IP of VM>; };
   b. Add IPs to be allowed to query the DNS in the allow-query as follows:

```
allow-query { any; }; //where 'any' specifies that any IP can query the DNS
```

   c. Define acl as follows:
      acl <name> { <IP/Subnet>; };
   d. Define views as follows:

```
view  "<view-name>" {
      //allows access only to source IPs defined by acl
match-clients { <acl name>; };
.....
.....
};
```

   e. Define zones inside each view. All the zones must reside in views:

```
view  "<view-name>" {
```

```
      match-clients { <acl-name>; };
```

1. zone "<zone-name>" IN {
   //specifies whether the zone is master/ slave
   type master;
   //specifies the file to be referred for zone definition
   file "<zone-file name>";
   allow-update { none; };
   };

   //lists the root domain name servers for local network
   zone "." IN {
      type hint;
      file "named.ca";
   };
};

3. Create zone files for each zone in /var/named/ directory as follows:

```
$TTL 86400
@   IN  SOA     <zone-ns> root.<zone> (
        2011071001  ;Serial
        3600        ;Refresh
        1800        ;Retry
        604800      ;Expire
        86400       ;Minimum TTL
)
@     IN  NS         <zone-ns>.
@     IN  A           <IP of zone-ns>
tenant       IN  A   <IP of tenant DNS>
```

4. Enable and initiate the DNS service as follows:
   systemctl enable named
   systemctl start named

   If errors are returned, use the following commands to know more about errors:
   journalctl -xe

5. Allow DNS service through firewall using the following commands and restart it:

```
firewall-cmd --permanent --add-port=53/tcp
      firewall-cmd --permanent --add-port=53/udp
```

```
Firewall-cmd --reload
```

6. Configure permissions as follows:
   chgrp named -R /var/named
   chown -v root:named /etc/named.conf
   restorecon -rv /var/named
   restorecon /etc/named.conf

7. Verify the named.conf configuration using the following commands:
   named-checkconf /etc/named.conf
   It returns nothing if there are no errors. If it returns error, check named.conf for errors.

8. Verify the zone configuration using the following commands:
   Named-checkzone <zone-ns> /var/named/<zone-filename>
   It returns OK if no error is found and loads the zone file

9. Add IP of DNS server in network interface file in /etc/sysconfig/network-scripts/<interface> as follows:
   DNS="<IP of DNS VM>"

10. Edit the IP of nameserver in /etc/resolv.conf file to add IP of DNS VM.
    Nameserver <IP of DNS VM>

Setting up high availability on VPC DNS:

1. Install the following on the servers on which HA needs to be configured.
   a. yum install gcc kernel-headers kernel-devel
   b. yum install keepalived
2. Edit the /etc/keepalived/keepalived.conf file on each server.
   a. On the primary server use a config similar to the below:

```
global_defs {
   notification_email {
     sysadmin@mydomain.com
     support@mydomain.com
   }
   notification_email_from lb1@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VI_1 {
    state MASTER
```

```
    interface eth1
    virtual_router_id 51
    priority 101
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.10.121
    }
}
```

b. On the secondary use a config similar to the below:

```
global_defs {
   notification_email {
     sysadmin@mydomain.com
     support@mydomain.com
   }
   notification_email_from lb2@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VI_1 {
    state MASTER
    interface eth1
    virtual_router_id 51
    priority 80
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.219.20
    }
}
```

Ensure both the servers listen to DNS requests on the Virtual IP assigned to them

3. Use “ip addr show ethx” and observe that only one of the instance has the virtual IP configured on it.

Setting up a firewall:

- For implementing a firewall which limits the rate of incoming requests from a particular IP we use the below rules at the hypervisor TLD DNS server.

  iptables -t mangle -A PREROUTING -p icmp -m hashlimit --hashlimit-name icmp --hashlimit-mode srcip --hashlimit 30/minute --hashlimit-burst 1 -j ACCEPT

  iptables -t mangle -A PREROUTING -p icmp -j DROP

- We have tested our implementation with icmp ping packets with the below rules

  iptables -t mangle -A PREROUTING -p tcp -m hashlimit --hashlimit-name tcp --hashlimit-mode srcip --hashlimit 3/second --hashlimit-burst 5 -j ACCEPT

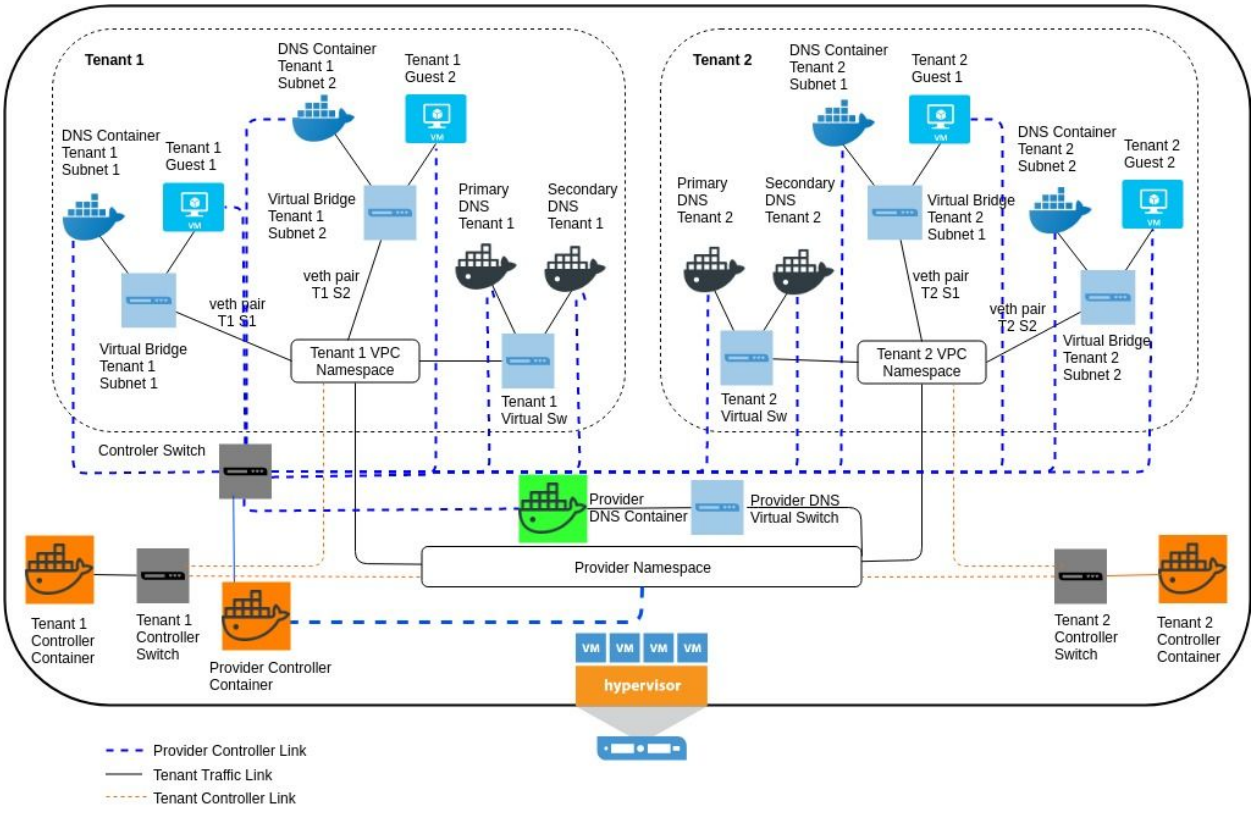
  iptables -t mangle -A PREROUTING -p tcp -j DROP

4.1.4 Developer Guide:

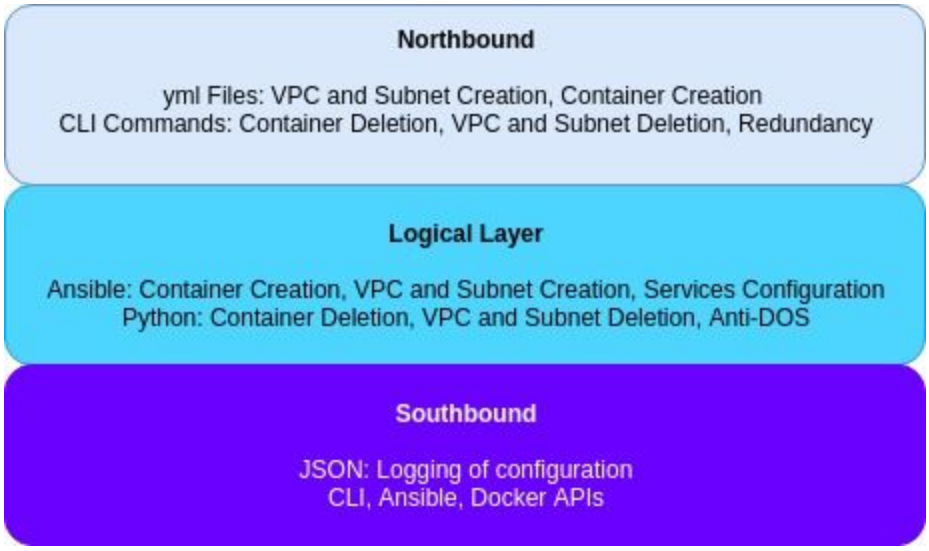
The developer guide related to VM architecture, VPC, subnet and instance creation is mentioned in **README_VM.pdf**

4.2 Architecture using Containers

4.2.1 Container Implementation Architecture



4.2.2 Functional Diagram of Containerized Architecture



- Northbound: Input Collection Unit
    - Client uploads yml files with VPC, subnet and DNS, server instance creation files.
    - In case of deletion, user executes commands on CLI interface provided in the solution
- Logical Layer: Execution Unit

  We have following logical layers in the solution to take care of tasks mentioned below:
    - Ansible: Playbooks are updated with variables with the help of user input and executed for creation of VPC, DNS containers, subnets and configuring DNS services in hypervisor and tenant DNS containers
    - Python: Scripts are being executed to provide protection against DOS attacks, deletion of subnets, deletion of VPCs and containers.
- Southbound: Logging and CLI Output
    - CLI: It shows the output of the scripts being run showing the status of the provisioning/ deprovisioning of the infrastructure and services

4.2.3 User Guide:

Everything is automated in the containerized architecture environment with Kube dns configuration.
The user only needs to set up the high availability and DOS attack configurations

Setting up high availability on VPC DNS:

4. Install the following on the servers on which HA needs to be configured.
    a. yum install gcc kernel-headers kernel-devel
    b. yum install keepalived
5. Edit the /etc/keepalived/keepalived.conf file on each server.
    a. On the primary server use a config similar to the below:

```
global_defs {
   notification_email {
     sysadmin@mydomain.com
     support@mydomain.com
   }
   notification_email_from lb1@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VI_1 {
    state MASTER
    interface eth1
    virtual_router_id 51
    priority 101
    advert_int 1
```

```
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.10.121
    }
}
```

b. On the secondary use a config similar to the below:

```
global_defs {
   notification_email {
     sysadmin@mydomain.com
     support@mydomain.com
   }
   notification_email_from lb2@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VI_1 {
    state MASTER
    interface eth1
    virtual_router_id 51
    priority 80
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.219.20
    }
}
```

Ensure both the servers listen to DNS requests on the Virtual IP assigned to them

6. Use “ip addr show ethx” and observe that only one of the instance has the virtual IP configured on it.

Setting up a firewall:

- For implementing a firewall which limits the rate of incoming requests from a particular IP we use the below rules at the hypervisor TLD DNS server.

iptables -t mangle -A PREROUTING -p icmp -m hashlimit --hashlimit-name icmp --hashlimit-mode srcip --hashlimit 30/minute --hashlimit-burst 1 -j ACCEPT

iptables -t mangle -A PREROUTING -p icmp -j DROP

- We have tested our implementation with icmp ping packets with the below rules

  iptables -t mangle -A PREROUTING -p tcp -m hashlimit --hashlimit-name tcp --hashlimit-mode srcip --hashlimit 3/second --hashlimit-burst 5 -j ACCEPT

  iptables -t mangle -A PREROUTING -p tcp -j DROP

4.2.4 Developer Guide

The client guide and developer guide are both combined and explained in **README_Containers.pdf**. (which is attached here)

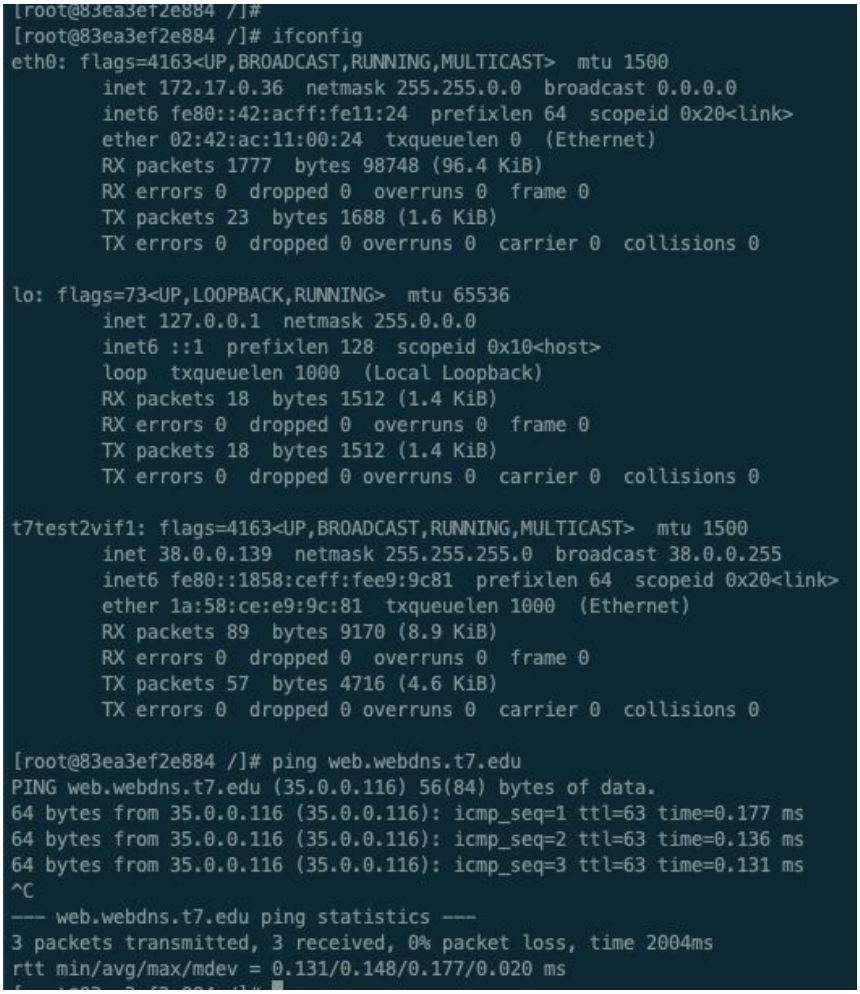
5 Kubedns Inspired Features:

5.1 Kubedns DNS Services communication

We created two services for tenant : test and web and we tried to ping from test server to web1 server and it got resolved and the ping happened successfully.

```
services:
   -  web_service:
          - webserver1
          - webserver2
   -  test_service:
          - testserver1
        -  testserver2
```

(This shows east to west communication which is fully automated.)



5.2 High Availability and Automatic Recovery of Containers:

We provide automatic recovery from arbitrary exits in our design by running a python script for each tenant's containers. This script executes every T seconds (10 by default) and checks the status of the executing containers. If it finds any of them in the stopped state, i.e. the user has accidentally exited from them, it starts those containers again.

In the screenshot above, we can see that the container tc1 is executing initially. We store its id in a file named sample_continers.txt and pass it as an argument to the handle_exits.py file. The script fetches the container ids stored in the file, every 10 seconds, it checks whether any of the containers with the container Ids in the file are in the stopped state. If yes, it starts them.
Below is the content of the script.

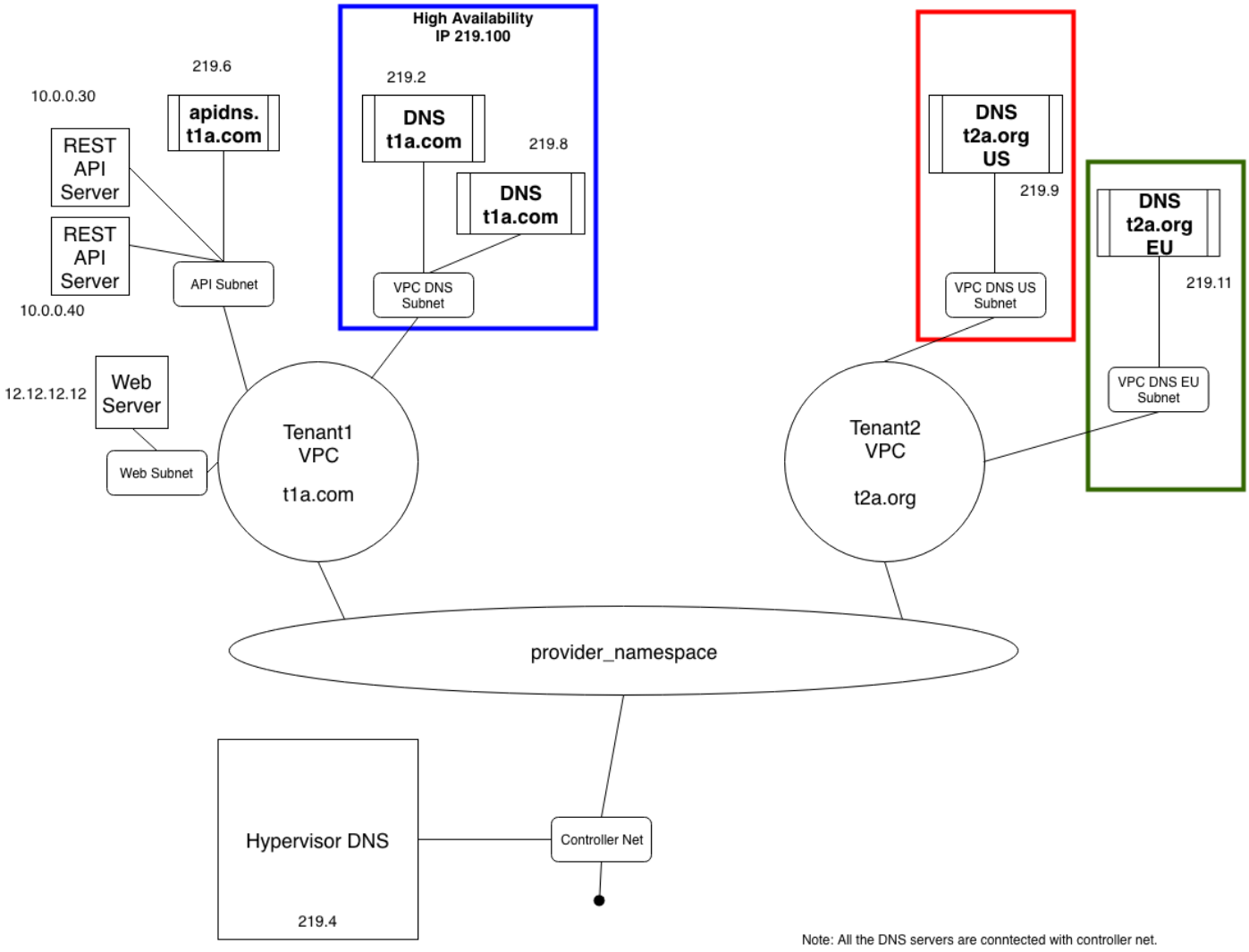
```
import os
import docker
import sys
from time import sleep

dclient = docker.from_env()
if len(sys.argv) < 2:
    print "Insufficient arguments"
    sys.exit()

#Fetch container ids from input file
cid_file = open(sys.argv[1],"r")
cids = cid_file.read()
cids = [t.strip() for t in cids.split()]
```

```
#Loop
while True:
    sleep(10)
    #Get active containers
    curr_containers = docker.Client().containers(all=True)

    #Check if any with it's id present in the cids file is in stopped state
    for container in curr_containers:
        if "Exited" in container["Status"]:
            for cid in cids:
                if cid in container["Id"]:
                    #Start if it is in stopped state
                    print "Starting container with id ",container["Id"]
                    docker.Client().start(container)
```

Automatic VPC zone configuration - Automating the complete DNS setup is not feasible using ansible as the places where views, ACLs are added is not static and dependent on the order in which they are added. Kube DNS allows autoconfiguration of the A records of a newly created subnet. We try to provide similar functionality by appending the A records of the newly created subnet in a tenant's VPC to the namespace level DNS sever's "forward" file. As the tenant level policies are defined at the namespace level, we need not check them again at the subnet level. The subnet will only serve requests for the servers present in the particular subnet, which makes configuration simpler than the top level and namespace level servers.

6. Evaluation Section

6.1 Architecture using VM



6.1.1 Hypervisor DNS Setup

- Command: sudo ansible-playbook provider-configuration/hv_dns.yml
- Expected Output:
  Ansible output should run successfully
- Resolv.conf of Hypervisor DNS

```
[root@localhost ~]# cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
nameserver 192.168.219.20
[root@localhost ~]#
```

- Named.conf for Hypervisor DNS Container

```
acl US {
                192.168.154.0/24; };

acl EU {        192.168.219.0/24;
                192.168.122.0/24; };
acl others {    0.0.0.0/24;  };

view "others" {
        match-clients { others; };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };
};

view "US" {
        match-clients { US; };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forward.com";
```

```
                allow-update { none; };
                allow-transfer { any; };
        };
};

view  "EU" {
        match-clients { EU; };

        zone "org" IN {
                type master;
                file "forwardeu.org";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "." IN {
                type hint;
                file "named.ca";
        };
};
```

- Forward.com of HV DNS

```
[root@localhost named]# cat forward.com
$TTL 86400
@   IN  SOA    com. root.com. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
@       IN  NS          t1a.com.
@       IN  A           192.168.219.32
web.t1a IN  NS          t1a.com.
api.apidns.t1a  IN      NS      apidns.t1a.com.
apidns.t1a      IN  NS          t1a.com.
api.t1a         IN      CNAME   api.apidns.t1a.com.
t1a     IN  A           192.168.219.2
[root@localhost named]#
```

- Ip address of HV DNS

```
[root@localhost ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.219.20  netmask 255.255.255.0  broadcast 192.
168.219.255
        ether 52:54:00:a7:9d:91  txqueuelen 1000  (Ethernet)
        RX packets 13318508  bytes 627137840 (598.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 190978  bytes 21653059 (20.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.122.39  netmask 255.255.255.0  broadcast 192.
168.122.255
        ether 52:54:00:e3:7a:f3  txqueuelen 1000  (Ethernet)
        RX packets 1515044  bytes 981414982 (935.9 MiB)
        RX errors 0  dropped 33  overruns 0  frame 0
        TX packets 354704  bytes 1293171609 (1.2 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 11383  bytes 1101102 (1.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11383  bytes 1101102 (1.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6.1.2 Creation of VPC

It is the same as in section 6.2.2

6.1.2.1 Creation of VPC DNS Subnet

It is the same as in section 6.2.2.1

6.1.2.2 Creation of VPC DNS

- Command: sudo ansible-playbook
- Input for creating VPC DNS

```

```

Ansible-playbook successfully run

Resolv.conf for VPC DNS

```
# Generated by NetworkManager
nameserver 192.168.219.2
```

Zone for VPC DNS

```
        managed-keys-directory "/var/named/dynamic";

        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
};

logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

zone "." IN {
        type hint;
        file "named.ca";
};

zone "t1a.com" IN {
        type master;
        file "forward.t1a";
        allow-update { none; };
        allow-query { any; };
};

zone "219.168.192.in-addr.arpa" IN {
        type master;
        file "reverse.t1a";
        allow-update { none; };
        allow-query { any; };
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

Ip config of VPC DNS

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.219.2  netmask 255.255.255.0  broadcast 192.168.219.25
5
        ether 52:54:00:5a:ef:6b  txqueuelen 1000  (Ethernet)
        RX packets 7392  bytes 621164 (606.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3170  bytes 381350 (372.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 25.0.0.173  netmask 255.255.255.0  broadcast 25.0.0.255
        inet6 fe80::27a5:3e20:62a2:f1ac  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:d5:28:7f  txqueuelen 1000  (Ethernet)
        RX packets 178  bytes 19258 (18.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 572  bytes 47544 (46.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 97  bytes 8997 (8.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 97  bytes 8997 (8.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Forward.com for VPC DNS

```
$TTL 86400
@   IN  SOA     t1a.com. root.t1a.com. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        800       ;Minimum TTL
)
@       IN  NS          apidns.t1a.com.
@       IN  NS          web.t1a.com.
@       IN  A           192.168.219.2
api.apidns      IN  NS          apidns.t1a.com.
apidns  IN  A           192.168.219.6
web     IN  A           12.12.12.12
~
~
```

Nslookup from VPC DNS for t1a.com

```
[root@localhost ~]# nslookup t1a.com
Server:         192.168.219.2
Address:        192.168.219.2#53

Name:   t1a.com
Address: 192.168.219.2

[root@localhost ~]#
```

NSlookup from HV DNS for t1a.com

```
[root@localhost ~]# nslookup t1a.com
Server:         192.168.219.20
Address:        192.168.219.20#53

Name:   t1a.com
Address: 192.168.219.2

[root@localhost ~]#
```

6.1.3 Creation of Subnet

- It's the same as 6.2.3

6.1.4 Creation of VM DNS at Subnet Level

Resolv.conf

```
; generated by /usr/sbin/dhclient-script
search example.org
nameserver 192.168.219.6
~
```

Ipconfig of Subnet DNS

```
[root@localhost ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.219.6  netmask 255.255.255.0  broadcast 192.168.219.25
5
        ether 52:54:00:31:d2:13  txqueuelen 1000  (Ethernet)
        RX packets 13180778  bytes 610405630 (582.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 52776  bytes 6407995 (6.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.45  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::5796:7ea:89bc:5f7c  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:ee:c0:84  txqueuelen 1000  (Ethernet)
        RX packets 6793  bytes 559393 (546.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4681  bytes 1202309 (1.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1212  bytes 145120 (141.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1212  bytes 145120 (141.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@localhost ~]#
```

Forward file of subnet DNS

```
$TTL 86400
@   IN  SOA     api.apidns.t1a.com. root.apidns.t1a.com. (
        2011071001  ;Serial
        3600        ;Refresh
        1800        ;Retry
        604800      ;Expire
        86400       ;Minimum TTL
)
@       IN  NS          api.apidns.t1a.com.
@       IN  A           192.168.219.6
;@       IN  A           10.0.0.30
;@       IN  A           10.0.0.40
api     IN  A           10.0.0.30
api     IN  A           10.0.0.40
;*      IN  A           10.0.0.40
```

6.1.5 Outputs

Nslookup from hv dns for api.apidns.t1a.com and api.t1a.com:

```
[root@localhost ~]# nslookup api.apidns.t1a.com
Server:         192.168.219.20
Address:        192.168.219.20#53

Non-authoritative answer:
Name:   api.apidns.t1a.com
Address: 10.0.0.40
Name:   api.apidns.t1a.com
Address: 10.0.0.30

[root@localhost ~]# nslookup api.t1a.com
Server:         192.168.219.20
Address:        192.168.219.20#53

api.t1a.com     canonical name = api.apidns.t1a.com.
Name:   api.apidns.t1a.com
Address: 10.0.0.30
Name:   api.apidns.t1a.com
Address: 10.0.0.40
```

Nslookup from HV DNS for t1a.com

```
[root@localhost ~]# vi /etc/resolv.conf
[root@localhost ~]# nslookup t1a.com
Server:         192.168.219.20
Address:        192.168.219.20#53

Name:   t1a.com
Address: 192.168.219.2

[root@localhost ~]#
```

6.1.6 Feature implementation:

6.1.6.1 Feature: VM Content based routing

We created two api servers with api1 having IP address as 10.0.0.40 and api2 as 10.0.0.30 We can see in the below screenshot that the content based load balancing is happening in round-robin fashion.

```
[root@localhost ~]# nslookup api.t1a.com
Server:         192.168.219.20
Address:        192.168.219.20#53

api.t1a.com     canonical name = api.apidns.t1a.com.
Name:   api.apidns.t1a.com
Address: 10.0.0.40
Name:   api.apidns.t1a.com
Address: 10.0.0.30

[root@localhost ~]# nslookup api.t1a.com
Server:         192.168.219.20
Address:        192.168.219.20#53

api.t1a.com     canonical name = api.apidns.t1a.com.
Name:   api.apidns.t1a.com
Address: 10.0.0.30
Name:   api.apidns.t1a.com
Address: 10.0.0.40

[root@localhost ~]#
```

6.1.6.2 Feature: VM Geographical based routing

Changed the zone in named.conf of HV DNS with US zone returning for IP 192.168.219.0 /24

ACL screenshot for US zone

```
acl US {      192.168.219.0/24;
                192.168.154.0/24; };

acl EU {
                192.168.122.0/24; };
acl others {    0.0.0.0/24;  };

view "others" {
        match-clients { others; };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };
};

view "US" {
        match-clients { US; };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };
};

view  "EU" {
        match-clients { EU; };

        zone "org" IN {
                type master;
                file "forwardeu.org";
                allow-update { none; };
```

Forward.com

```
$TTL 86400
@   IN  SOA    org. root.org. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
@       IN  NS          t2a.org.
@       IN  A           192.168.219.42
web.t2a IN  NS          t2a.org.
t2a     IN  A           192.168.219.42
```

Nslookup for t2a.org

```
[root@localhost ~]# nslookup t2a.org
Server:         192.168.219.20
Address:        192.168.219.20#53

Name:   t2a.org
Address: 192.168.219.32
```

Changed the zone in named.conf of HV DNS with EU zone returning for IP 127.0.0.0/8

ACL screenshot

```
acl US {
                192.168.154.0/24; };

acl EU {        192.168.219.0/24;
                192.168.122.0/24; };
acl others {    0.0.0.0/24;  };

view "others" {
        match-clients { others; };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };
};

view "US" {
        match-clients { US; };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };
};
```

Forward table for EU zone:

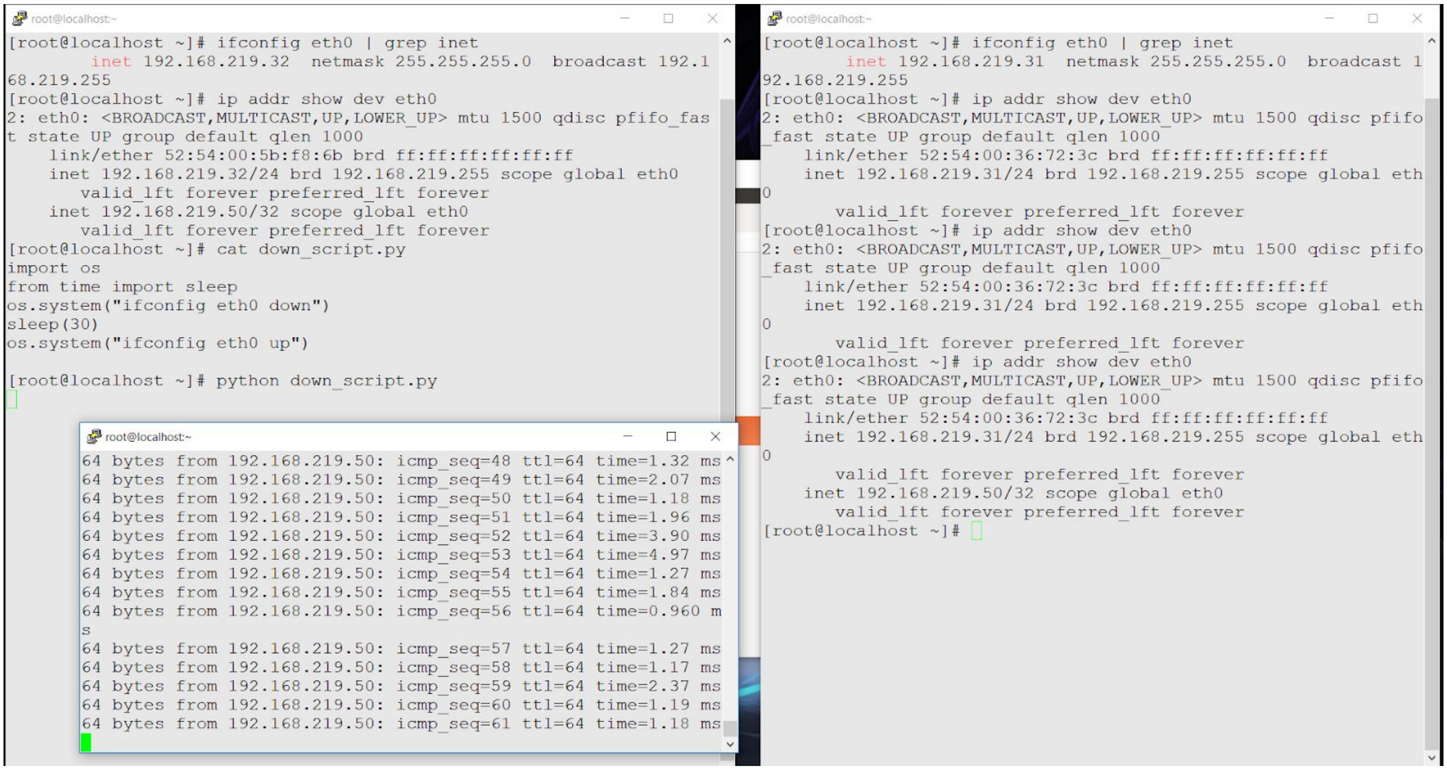
Nslookup for t2a.org

```
[root@localhost ~]# nslookup t2a.org
Server:         192.168.219.20
Address:        192.168.219.20#53

Name:   t2a.org
Address: 192.168.219.42

[root@localhost ~]#
```

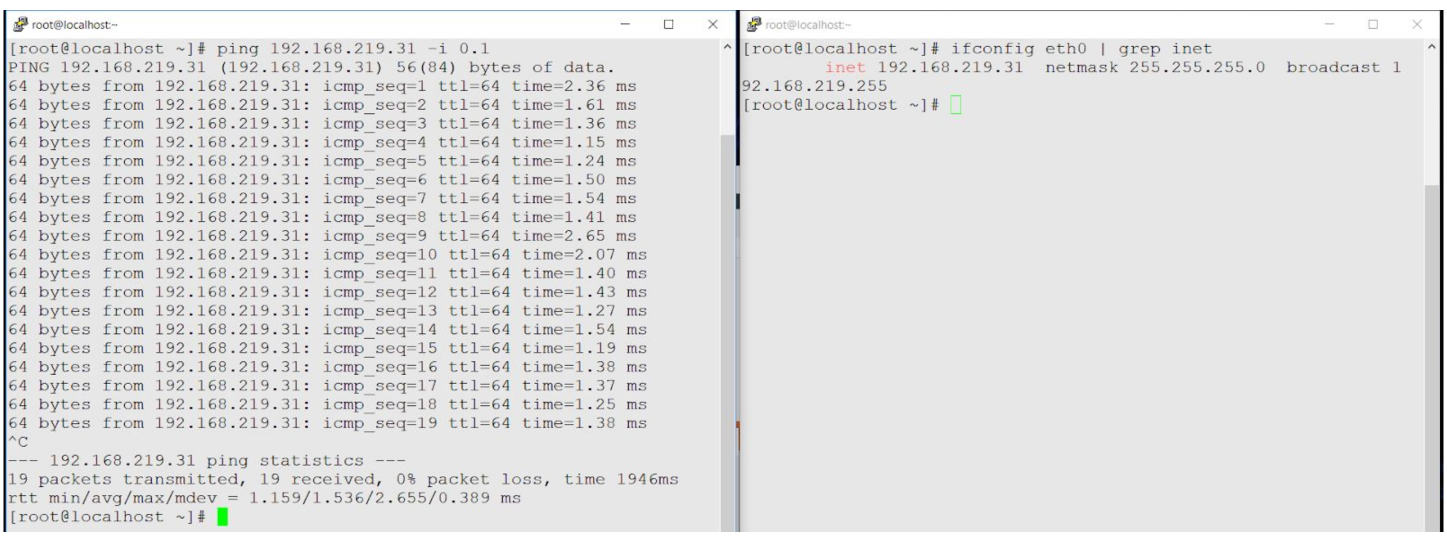
6.1.6.3. VRRP (High Availability)

The above screenshot of high availability using keepalived in VMs. At the topmost lines, we can observe that both of them are in the 192.168.219.0/24 subnet. They both internally are configured to serve the 192.168.219.50 address. The ".32" has a priority of 100, whereas ".31" has a priority of 80. The script down_script.py turns an interface down for 30 seconds. As we can observe, when the script is executed, there

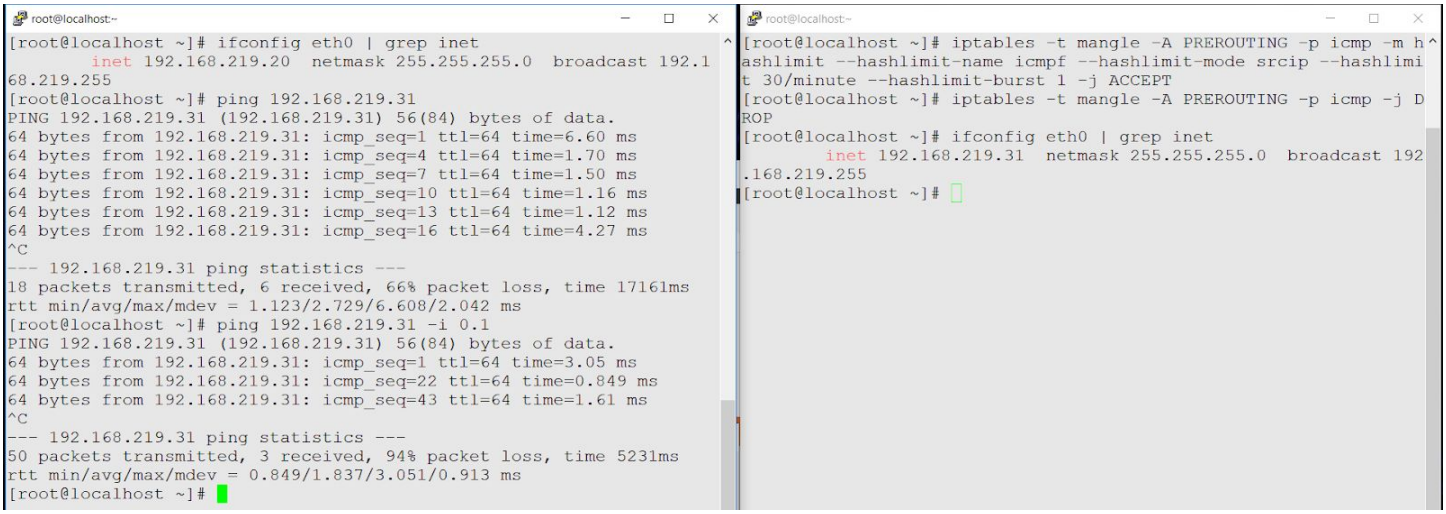
is almost no loss of ping packets (in the bottom window). We can observe that the secondary i.e the node with lower priority picks up the IP as soon as it detects that the primary is down.



6.1.6.4 DOS Attack



The above screenshot displays a ping between 192.168.219.31 without DOS rues in effect. It can be observed, that there is no loss of packets.



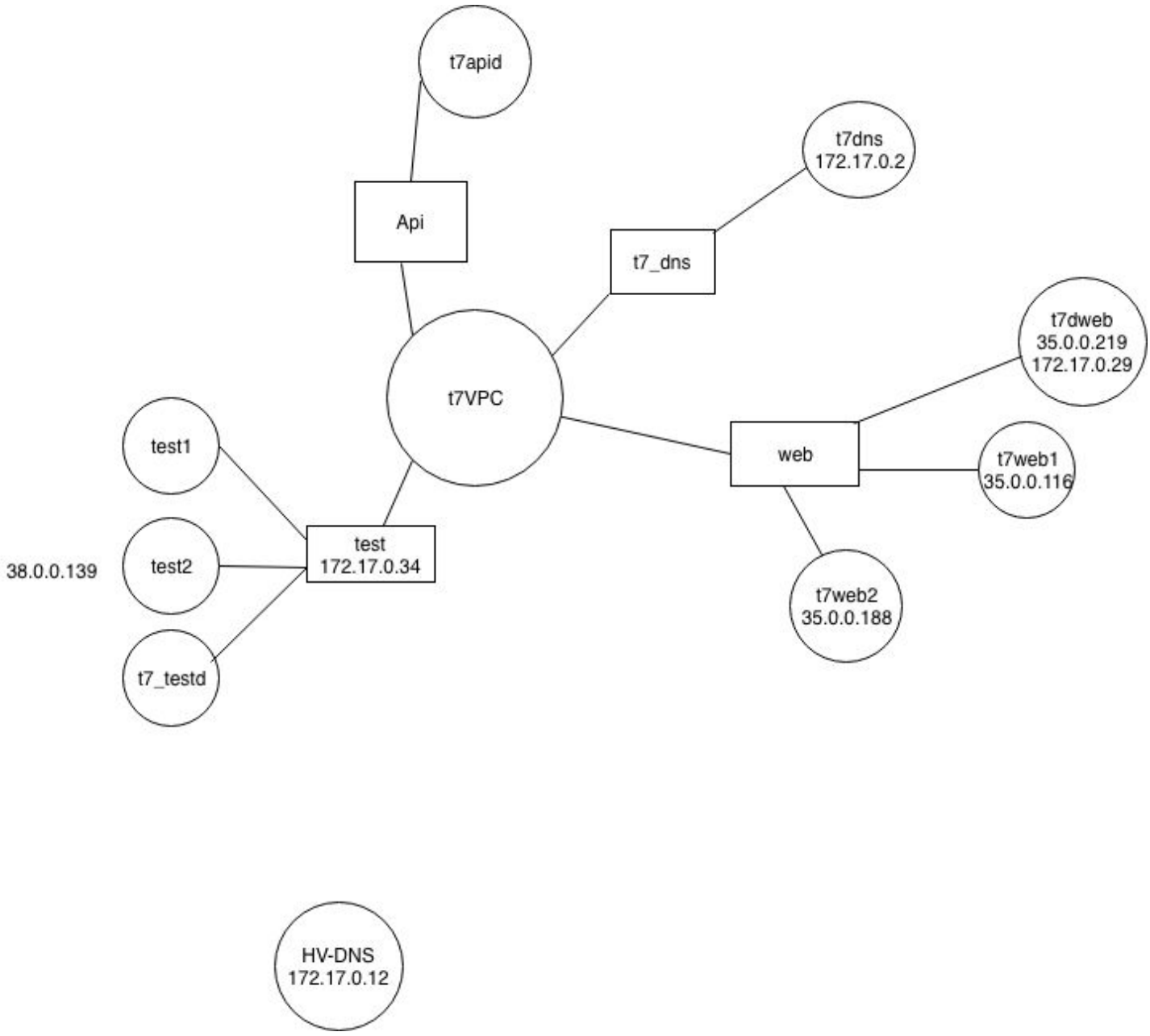
After placing a rule which limits the traffic to 30 packets/minute, we can observe that the number of ping responses drops drastically. In the destination, we have added a rule to track and hash connections based on source IP. This can be extended to hosts internal to the VPC and external to the VPC. Different throughput limits can be set for trusted and untrusted hosts.

6.1.6.5 Controller SSH inside tenant DNS VMs

We have given a management feature for the tenant to login to any of its DNS vm inside its own vpc by logging into one controller vm . Here, the tenant 1 has a controller vm with ip address 192.168.219.13 and the tenant is logging into its DNS which has ip address 25.0.0.173

```
[root@localhost ~]# ssh root@192.168.219.13
root@192.168.219.13's password:
Last login: Wed Nov 28 23:30:25 2018 from 192.168.219.20
[root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:7e:a1:a6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.219.13/24 brd 192.168.219.255 scope global noprefixroute dynamic eth0
       valid_lft 409sec preferred_lft 409sec
    inet6 fe80::2d19:961a:175:31cb/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
    inet6 fe80::9134:ae8d:f9cb:9bad/64 scope link tentative noprefixroute dadfailed
       valid_lft forever preferred_lft forever
    inet6 fe80::7ce5:9488:6904:61e6/64 scope link tentative noprefixroute dadfailed
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:7e:03:f0 brd ff:ff:ff:ff:ff:ff
    inet 82.0.0.45/24 brd 82.0.0.255 scope global dynamic eth1
       valid_lft 39908sec preferred_lft 39908sec
[root@localhost ~]# ssh root@25.0.0.173
root@25.0.0.173's password:
Last login: Fri Dec  7 22:04:46 2018 from 82.0.0.45
[root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:5a:ef:6b brd ff:ff:ff:ff:ff:ff
    inet 192.168.219.2/24 brd 192.168.219.255 scope global eth0
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:d5:28:7f brd ff:ff:ff:ff:ff:ff
    inet 25.0.0.173/24 brd 25.0.0.255 scope global noprefixroute dynamic eth1
       valid_lft 39579sec preferred_lft 39579sec
    inet6 fe80::27a5:3e20:62a2:f1ac/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
[root@localhost ~]#
```

6.2 Architecture using Containers



6.2.1

- Command: sudo ansible-playbook provider-configuration/hv_dns.yml
- input: nothing required as this is provider DNS
- Expected Output:
- Ansible script should run successfully

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/DNS-as-a-service$ sudo ansible-playbook provider_configuration/hv_dns.yml
[sudo] password for ece792:
 [WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'


PLAY [localhost] ***************************************************************************************************************

TASK [debug] *******************************************************************************************************************
ok: [localhost] => {
    "msg": {
        "provider": {
            "dns_name": "hv_dns",
            "publish_port": 79
        }
    }
}

TASK [Install required packages for docker] ************************************************************************************
ok: [localhost]

TASK [Create docker image if not exists] ***************************************************************************************
ok: [localhost]

TASK [Create shared volume in Hypervisor] **************************************************************************************
changed: [localhost] => (item=provider)

TASK [Create and start container for dns if already not present] ***************************************************************
changed: [localhost] => (item=provider)

TASK [Start the named] *********************************************************************************************************
changed: [localhost] => (item=provider)

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=6    changed=3    unreachable=0    failed=0
```

- Resolv.conf of Hypervisor DNS

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/DNS-as-a-service$ docker exec -it hv_dns bash
[root@d6297b417a48 /]# clear
[root@d6297b417a48 /]#
[root@d6297b417a48 /]# cat /etc/resolv.conf
nameserver 127.0.0.1
```

- Named.conf for Hypervisor DNS Container

```
acl US {        172.16.0.0/12; };
acl EU {        192.168.154.0/24; };
acl others {    0.0.0.0/0; };

view "others" {
        match-clients { others; };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "edu" IN {
                type master;
                file "forward.edu";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "." IN {
                type hint;
                file "named.ca";
        };
};

view "US" {
        match-clients { US; };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "edu" IN {
                type master;
                file "forward.edu";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "us" IN {
                type master;
                file "forward.us";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "." IN {
                type hint;
                file "named.ca";
        };
};

view  "EU" {
        match-clients { EU; };

        zone "org" IN {
                type master;
                file "forward.org";
                allow-update { none; };
                allow-transfer { any; };
        };
```

- Forward.com of HV DNS

```
[root@d6297b417a48 /]# cat /named/var/forward.com
$TTL 86400
@   IN  SOA    com. root.com. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
```

- IP address of HV DNS

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo docker exec -it hv_dns bash
[root@d6297b417a48 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
    link/gre 0.0.0.0 brd 0.0.0.0
3: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
2440: eth0@if2441: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:0c brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.12/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:c/64 scope link
       valid_lft forever preferred_lft forever
```

6.2.2 Creation of VPC

Ansible-playbook for VPC Creation

- Command: sudo ansible-playbook create_vpc.yml
- Input: vpc_vars.yml

```
---
vpcs:
  t7:   # Tenant Name
    vpc_name: t7vpc
    vpc_gateway: 104.0.0.2/24
    provider_gateway: 104.0.0.1/24
    provider_gateway_ip: 104.0.0.1
```

- Expected Output:
    - Ansible-playbook should run successfully

```
TASK [Debug new database] ******************************************************
ok: [localhost] => {
    "msg": {
        "t7vpc": {
            "t7": {
                "provider_gateway": "104.0.0.1/24",
                "provider_gateway_ip": "104.0.0.1",
                "vpc_gateway": "104.0.0.2/24",
                "vpc_name": "t7vpc"
            }
        }
    }
}

TASK [Write db back to file] ***************************************************
changed: [localhost] => (item=t7)

PLAY RECAP *********************************************************************
localhost                  : ok=26   changed=10   unreachable=0    failed=0
```

- Database for new tenant should be written

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/DNS-as-a-service$ sudo cat ../tenants/t7/vpc_db.json
{
    "t7vpc": {
        "t7": {
            "provider_gateway": "104.0.0.1/24",
            "provider_gateway_ip": "104.0.0.1",
            "vpc_gateway": "104.0.0.2/24",
            "vpc_name": "t7vpc"
        }
    }
```

- Namespace created and should be connected with provider_namespace

```
}ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/DNS-as-a-service$ sudo ip netns exec t7vpc_ns bash
root@ece792-Standard-PC-i440FX-PIIX-1996:~/DNS-as-a-service# ifconfig
t7vpcvif1 Link encap:Ethernet  HWaddr aa:8e:bf:23:c4:63
          inet addr:104.0.0.2  Bcast:104.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a88e:bfff:fe23:c463/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:866 (866.0 B)  TX bytes:866 (866.0 B)
```

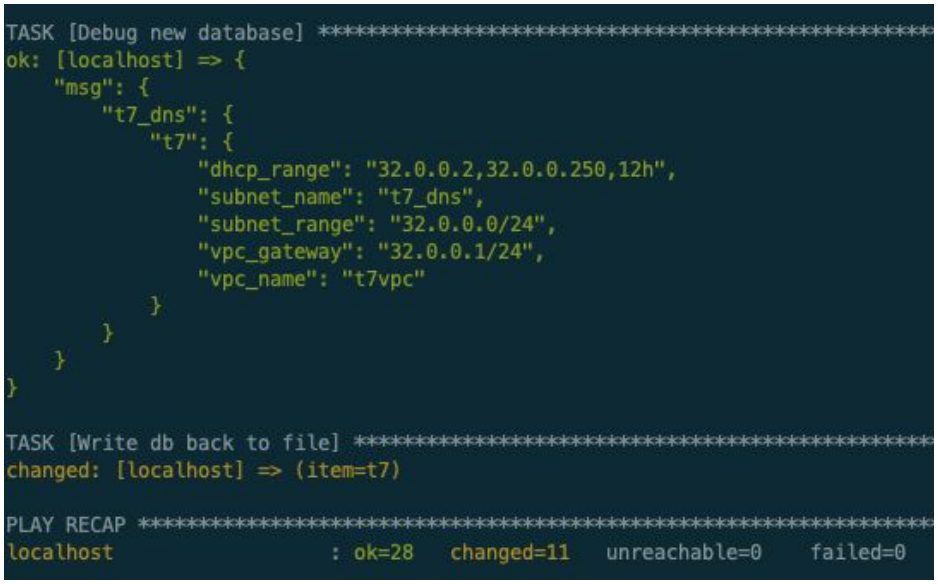
6.2.2.1 Creation of VPC DNS Subnet

- Command: sudo ansible-playbook create_subnet.yml
- Input for Creating DNS subnet:

```
---
subnets:
  t7: # Tenant name
    subnet_name: t7_dns
    vpc_name: t7vpc
    subnet_range: 32.0.0.0/24
```

```
    vpc_gateway: 32.0.0.1/24
    dhcp_range: 32.0.0.2,32.0.0.250,12h
```

- Expected Output: Ansible-playbook successfully run



6.2.2.2 Creation of VPC DNS

- Command: sudo ansible-playbook
- Input for creating VPC DNS

```
---
dns_containers:
    t7:
       dns_name: t7dns
       subnet_name: t7_dns
       publish_port: 107
       zone_name: t7.edu
       vpc_name: t7vpc
       level: vpc
       parent_dns_name: hv_dns
```

Ansible-playbook successfully run

```
TASK [Append new vpc to database file] ***********************************
ok: [localhost]

TASK [Debug new database] ************************************************
ok: [localhost] => {
    "msg": {
        "t7dns": {
            "t7": {
                "dns_name": "t7dns",
                "level": "vpc",
                "parent_dns_name": "hv_dns",
                "publish_port": 107,
                "subnet_name": "t7_dns",
                "vpc_name": "t7vpc",
                "zone_name": "t7.edu"
            }
        }
    }
}

TASK [Write db back to file] *********************************************
changed: [localhost] => (item=t7)

PLAY RECAP ***************************************************************
localhost                  : ok=48   changed=21   unreachable=0    failed=0
```

- Resolv.conf for VPC DNS

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/DNS-as-a-service$ docker exec -it t7dns bash
[root@71d762ed1b30 /]# cat /etc/resolv.conf
nameserver 172.17.0.12
```

- Zone for VPC DNS

```
zone "t7.edu" IN {
        type master;
        file "forwardt7.edu";
        allow-update { none; };
        allow-transfer { any; };
};

include "/named/etc/named.rfc1912.zones";
include "/named/etc/named.root.key";[root@71d762ed1b30 /]#
```

- IP config of VPC DNS

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo docker exec -it t7dns bash
[root@71d762ed1b30 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
    link/gre 0.0.0.0 brd 0.0.0.0
3: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
2447: eth0@if2448: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:2/64 scope link
       valid_lft forever preferred_lft forever
2450: t7dnsvif1@if2449: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether de:0c:b7:9e:53:c6 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 32.0.0.204/24 brd 32.0.0.255 scope global t7dnsvif1
       valid_lft forever preferred_lft forever
    inet6 fe80::dc0c:b7ff:fe9e:53c6/64 scope link
       valid_lft forever preferred_lft forever
```

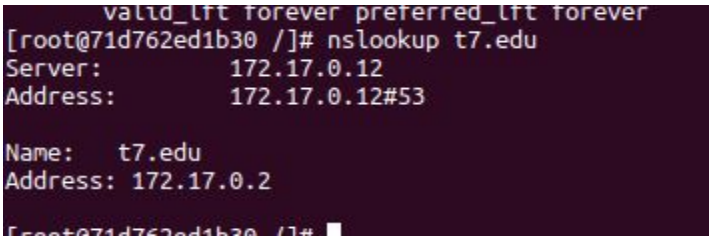
- Forward.com for VPC DNS

```
[root@71d762ed1b30 /]# cat /named/var/forwardt7.edu
$TTL 86400
@   IN  SOA    t7.edu. root.t7.edu. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
@       IN  NS          t7.edu.
@       IN  A           172.17.0.2
```

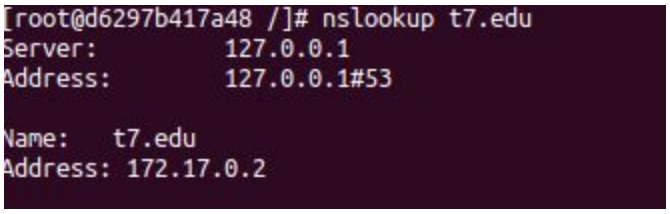
- Change in forward.com of HV DNS

```
[root@d6297b417a48 /]# cat /named/var/forward.edu
$TTL 86400
@   IN  SOA    edu. root.edu. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
@    IN  NS  t7.edu.
t7    IN  A   172.17.0.2
[root@d6297b417a48 /]#
```

- Nslookup from VPC DNS for t7.edu should return 172.17.0.2



- NSlookup from HV DNS for t7.edu should return 172.17.0.2



6.2.3 Creation of Subnet

- **Command:** sudo ansible-playbook create_subnet.yml
- **Input:** subnet_vars.yml file

```
---
subnets:
  t7: # Tenant name
    subnet_name: t7_web
    vpc_name: t7vpc
    subnet_range: 35.0.0.0/24
    vpc_gateway: 35.0.0.1/24
    dhcp_range: 35.0.0.2,35.0.0.250,12h
```

- Expected Output and actual output results shown in screenshot:
  - Ansible-playbook should run completely

```
            }
        },
        "t7_web": {
            "t7": {
                "dhcp_range": "35.0.0.2,35.0.0.250,12h",
                "subnet_name": "t7_web",
                "subnet_range": "35.0.0.0/24",
                "vpc_gateway": "35.0.0.1/24",
                "vpc_name": "t7vpc"
            }
        }
    }
}

TASK [Write db back to file] ***************************************************
changed: [localhost] => (item=t7)

PLAY RECAP *********************************************************************
localhost                  : ok=27   changed=10   unreachable=0    failed=0
```

- Newly created subnet should be written in database

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/DNS-as-a-service$ sudo cat ../tenants/t7/subnet_db.json
{
    "t7_api": {
        "t7": {
            "dhcp_range": "33.0.0.2,32.0.0.250,12h",
            "subnet_name": "t7_api",
            "subnet_range": "33.0.0.0/24",
            "vpc_gateway": "33.0.0.1/24",
            "vpc_name": "t7vpc"
        }
    },
    "t7_dns": {
        "t7": {
            "dhcp_range": "32.0.0.2,32.0.0.250,12h",
            "subnet_name": "t7_dns",
            "subnet_range": "32.0.0.0/24",
            "vpc_gateway": "32.0.0.1/24",
            "vpc_name": "t7vpc"
        }
    },
    "t7_web": {
        "t7": {
            "dhcp_range": "35.0.0.2,35.0.0.250,12h",
            "subnet_name": "t7_web",
            "subnet_range": "35.0.0.0/24",
            "vpc_gateway": "35.0.0.1/24",
            "vpc_name": "t7vpc"
        }
    }
}
```

- Should create L2 bridge “t7_web” and attach with namespace “t7vpc” and the vethpair endpoint with namespace will have dhcp server running.

6.2.4 Creation of DNS at Subnet Level

- Command: sudo ansible-playbook create_dns.yml
- Input file

```
---
dns_containers:
    t7:
       dns_name: t7dweb
       subnet_name: t7_web
       publish_port: 110
       zone_name: webdns.t7.edu
       vpc_name: t7vpc
       level: subnet
       endpoint_name: web
       parent_dns_name: t7dns
```

- Ansible Playbook successful run

```
TASK [Debug new database] ******************************************************
ok: [localhost] => {
    "msg": {
        "t7dapi": {
            "t7": {
                "dns_name": "t7dapi",
                "endpoint_name": "api",
                "level": "subnet",
                "parent_dns_name": "t7dns",
                "publish_port": 108,
                "subnet_name": "t7_dns",
                "vpc_name": "t7vpc",
                "zone_name": "apidns.t7.edu"
            }
        },
        "t7dns": {
            "t7": {
                "dns_name": "t7dns",
                "level": "vpc",
                "parent_dns_name": "hv_dns",
                "publish_port": 107,
                "subnet_name": "t7_dns",
                "vpc_name": "t7vpc",
                "zone_name": "t7.edu"
            }
        },
        "t7dweb": {
            "t7": {
                "dns_name": "t7dweb",
                "endpoint_name": "web",
                "level": "subnet",
                "parent_dns_name": "t7dns",
                "publish_port": 110,
                "subnet_name": "t7_web",
                "vpc_name": "t7vpc",
                "zone_name": "webdns.t7.edu"
            }
        },
        "t7webd": {
            "t7": {
                "dns_name": "t7webd",
                "endpoint_name": "web",
                "level": "subnet",
                "parent_dns_name": "t7dns",
                "publish_port": 109,
                "subnet_name": "t7_dns",
                "vpc_name": "t7vpc",
                "zone_name": "webdns.t7.edu"
            }
        }
    }
}

TASK [Write db back to file] ***************************************************
changed: [localhost] => (item=t7)

PLAY RECAP *********************************************************************
localhost                  : ok=58   changed=23   unreachable=0    failed=0
```

- Resolv.conf

```
[root@9ea4c21525d9 /]# cat /etc/resolv.conf
nameserver 172.17.0.2
```

- Ipconfig of Subnet DNS

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo docker exec -it t7dweb bash
[root@9ea4c21525d9 /]#  ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
    link/gre 0.0.0.0 brd 0.0.0.0
3: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
2501: eth0@if2502: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:1d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.29/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:1d/64 scope link
       valid_lft forever preferred_lft forever
2504: t7dwebvif1@if2503: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen
    link/ether ca:6c:c4:e2:af:8e brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 35.0.0.219/24 brd 35.0.0.255 scope global t7dwebvif1
       valid_lft forever preferred_lft forever
    inet6 fe80::c86c:c4ff:fee2:af8e/64 scope link
       valid_lft forever preferred_lft forever
[root@9ea4c21525d9 /]#
```

- Forward file of subnet DNS

```
[root@9ea4c21525d9 /]# cat /named/var/forwardwebdns.t7.edu
$TTL 86400
@   IN  SOA    webdns.t7.edu. root.webdns.t7.edu. (
        2011071001  ;Serial
        10         ;Refresh
        5         ;Retry
        10       ;Expire
        20        ;Minimum TTL
)
@       IN  NS          webdns.t7.edu.
@       IN  A           172.17.0.29
@   IN  NS      web.webdns.t7.edu.
```

- Changes in vpc dns

```
[root@71d762ed1b30 /]# cat /named/var/forwardt7.edu
$TTL 86400
@   IN  SOA    t7.edu. root.t7.edu. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
@       IN  NS          t7.edu.
@       IN  A           172.17.0.2
@    IN  NS  apidns.t7.edu.
api.apidns   IN  NS  apidns.t7.edu.
apidns    IN  A   172.17.0.13
@    IN  NS  webdns.t7.edu.
web.webdns   IN  NS  webdns.t7.edu.
webdns    IN  A   172.17.0.29
```

- Changes in hv dns

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/DNS-as-a-service$ docker exec -it hv_dns bash
[root@d6297b417a48 /]# cat /named/var/forward.
forward.com  forward.edu  forward.org  forward.us
[root@d6297b417a48 /]# cat /named/var/forward.edu
$TTL 86400
@   IN  SOA    edu. root.edu. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
@    IN  NS  t7.edu.
t7    IN  A   172.17.0.2
apidns.t7   IN  NS t7.edu.
api.apidns.t7           IN              NS              apidns.t7.edu.
api.t7     IN     CNAME         api.apidns.t7.edu.
webdns.t7   IN  NS t7.edu.
web.webdns.t7           IN              NS              webdns.t7.edu.
web.t7     IN     CNAME         web.webdns.t7.edu.
```

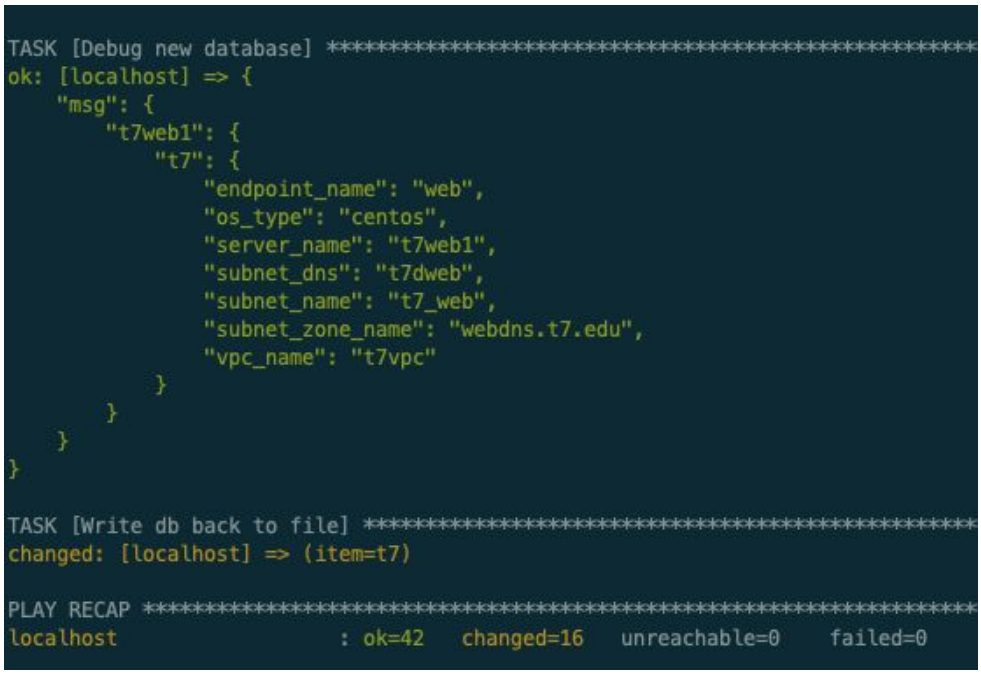
6.2.5 Creation of containerized server instance

- Command: sudo ansible-playbook create_server_instance.yml
- Input: instance_vars.yml

```
---
servers:
  t7: # Tenant Name
    server_name: t7api1
    subnet_name: t7_api
    vpc_name: t7vpc
    os_type: centos
    subnet_dns: t7dapi
    subnet_zone_name: apidns.t7.edu
    endpoint_name: api
```

- Expected Output:

Ansible Script should run successfully



- Resolv.conf

- Forward of subnet dns (A address

```
[root@9ea4c21525d9 /]# cat /named/var/forwardwebdns.t7.edu
$TTL 86400
@   IN  SOA    webdns.t7.edu. root.webdns.t7.edu. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
@       IN  NS          webdns.t7.edu.
@       IN  A           172.17.0.29
@   IN  NS      web.webdns.t7.edu.
web   IN  A  35.0.0.116
```

- Nslookup from subnet dns should return 35.0.0.116

```
[root@9ea4c21525d9 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
    link/gre 0.0.0.0 brd 0.0.0.0
3: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
2501: eth0@if2502: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:1d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.29/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:1d/64 scope link
       valid_lft forever preferred_lft forever
2504: t7dwebvif1@if2503: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether ca:6c:c4:e2:af:8e brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 35.0.0.219/24 brd 35.0.0.255 scope global t7dwebvif1
       valid_lft forever preferred_lft forever
    inet6 fe80::c86c:c4ff:fee2:af8e/64 scope link
       valid_lft forever preferred_lft forever
[root@9ea4c21525d9 /]# nslookup webdns.t7.edu
Server:         127.0.0.1
Address:        127.0.0.1#53

Name:   webdns.t7.edu
Address: 172.17.0.29

[root@9ea4c21525d9 /]# nslookup web.webdns.t7.edu
Server:         127.0.0.1
Address:        127.0.0.1#53

Name:   web.webdns.t7.edu
Address: 35.0.0.116
```

- Nslookup from vpc dns

```
[root@71d762ed1b30 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
    link/gre 0.0.0.0 brd 0.0.0.0
3: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
2447: eth0@if2448: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:2/64 scope link
       valid_lft forever preferred_lft forever
2450: t7dnsvif1@if2449: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether de:0c:b7:9e:53:c6 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 32.0.0.204/24 brd 32.0.0.255 scope global t7dnsvif1
       valid_lft forever preferred_lft forever
    inet6 fe80::dc0c:b7ff:fe9e:53c6/64 scope link
       valid_lft forever preferred_lft forever
[root@71d762ed1b30 /]# nslookup web.webdns.t7.edu
Server:         172.17.0.12
Address:        172.17.0.12#53

Non-authoritative answer:
Name:   web.webdns.t7.edu
Address: 35.0.0.116

[root@71d762ed1b30 /]# nslookup web.t7.edu
Server:         172.17.0.12
Address:        172.17.0.12#53

web.t7.edu      canonical name = web.webdns.t7.edu.
Name:   web.webdns.t7.edu
Address: 35.0.0.116

[root@71d762ed1b30 /]#
```

- Nslookup from hv dns should return 35.0.0.116

```
[root@d6297b417a48 /]# nslookup webdns.t7.edu
Server:         127.0.0.1
Address:        127.0.0.1#53

Non-authoritative answer:
Name:   webdns.t7.edu
Address: 172.17.0.27
Name:   webdns.t7.edu
Address: 172.17.0.29

[root@d6297b417a48 /]# nslookup web.webdns.t7.edu
Server:         127.0.0.1
Address:        127.0.0.1#53

Non-authoritative answer:
Name:   web.webdns.t7.edu
Address: 35.0.0.116

[root@d6297b417a48 /]# nslookup web.t7.edu
Server:         127.0.0.1
Address:        127.0.0.1#53

web.t7.edu      canonical name = web.webdns.t7.edu.
Name:   web.webdns.t7.edu
Address: 35.0.0.116
```

6.2.6 Feature Implementation:

6.2.6.1 Feature: Content based routing

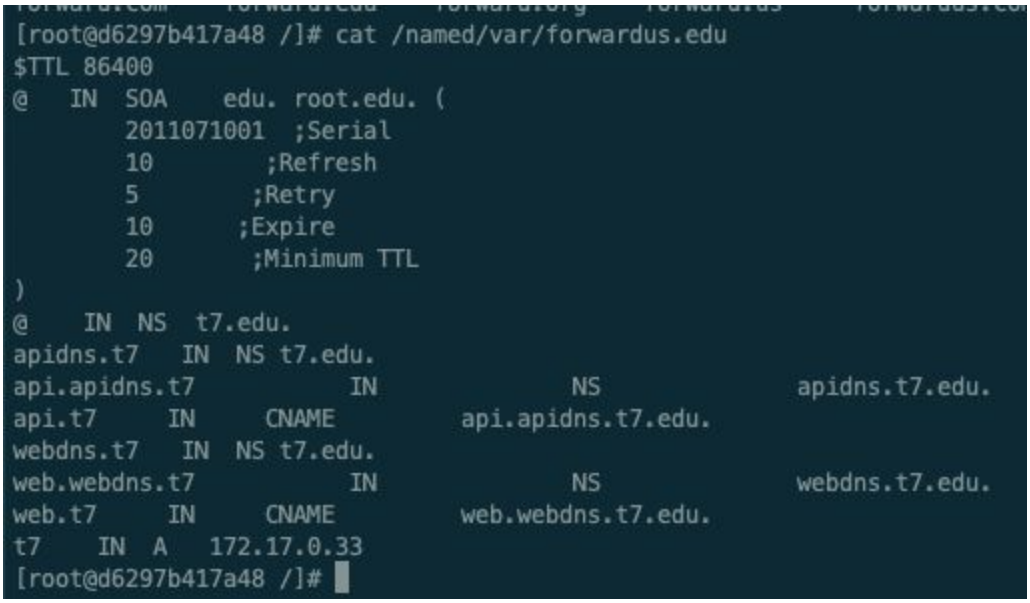
We created two erb servers with web1 having IP address as 35.0.0.116 and web2 as 35.0.0.188. We can see in the below screenshot that the content based load balancing is happening in round-robin fashion.

```
[root@9ea4c21525d9 /]# cat /named/var/forwardwebdns.t7.edu
$TTL 86400
@   IN  SOA    webdns.t7.edu. root.webdns.t7.edu. (
        2011071001  ;Serial
        10        ;Refresh
        5         ;Retry
        10       ;Expire
        20        ;Minimum TTL
)
@       IN  NS          webdns.t7.edu.
@       IN  A           172.17.0.29
@   IN  NS      web.webdns.t7.edu.
web     IN      A 35.0.0.116
web   IN  A  35.0.0.188
[root@9ea4c21525d9 /]#
```

```
[root@9ea4c21525d9 /]# nslookup web.webdns.t7.edu
Server:         172.17.0.2
Address:        172.17.0.2#53

Non-authoritative answer:
Name:   web.webdns.t7.edu
Address: 35.0.0.116
Name:   web.webdns.t7.edu
Address: 35.0.0.188

[root@9ea4c21525d9 /]# nslookup web.webdns.t7.edu
Server:         172.17.0.2
Address:        172.17.0.2#53

Non-authoritative answer:
Name:   web.webdns.t7.edu
Address: 35.0.0.188
Name:   web.webdns.t7.edu
Address: 35.0.0.116

[root@9ea4c21525d9 /]# nslookup web.webdns.t7.edu
Server:         172.17.0.2
Address:        172.17.0.2#53

Non-authoritative answer:
Name:   web.webdns.t7.edu
Address: 35.0.0.116
Name:   web.webdns.t7.edu
Address: 35.0.0.188
```

6.2.6.2 Feature: Geographical based routing

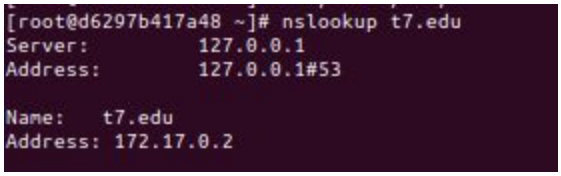
Changed the zone in named.conf of HV DNS with US zone returning for IP 127.0.0.0/8
Forwardus.edu



- ACL screenshot

```
acl US {       127.0.0.0/8; };
acl EU {        10.0.0.0/24; };

view "US" {
        match-clients { US; };

        zone "edu" IN {
                type master;
                file "forwardus.edu";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forwardus.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "us" IN {
                type master;
                file "forward.us";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "." IN {
                type hint;
                file "named.ca";
        };

};

view "EU" {
        match-clients { EU; };

        zone "edu" IN {
                type master;
                file "forward.edu";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "org" IN {
                type master;
                file "forward.org";
                allow-update { none; };
                allow-transfer { any; };
        };



        zone "." IN {
                type hint;
                file "named.ca";
        };

};
```

- Nslookup for t7.edu



Changed the zone in named.conf of HV DNS with EU zone returning for IP 127.0.0.0/8

- ACL screenshot

```
acl US {       10.0.0.0/12; };
acl EU {        127.0.0.0/8; };

view "US" {
        match-clients { US; };

        zone "edu" IN {
                type master;
                file "forwardus.edu";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forwardus.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "org" IN {
                type master;
                file "forwardus.org";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "us" IN {
                type master;
                file "forward.us";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "." IN {
                type hint;
                file "named.ca";
        };

};

view "EU" {
        match-clients { EU; };

        zone "edu" IN {
                type master;
                file "forward.edu";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "com" IN {
                type master;
                file "forward.com";
                allow-update { none; };
                allow-transfer { any; };
        };

        zone "org" IN {
                type master;
                file "forward.org";
                allow-update { none; };
                allow-transfer { any; };
        };



        zone "." IN {
                type hint;
                file "named.ca";
        };

};
```

- Forward table for EU zone:

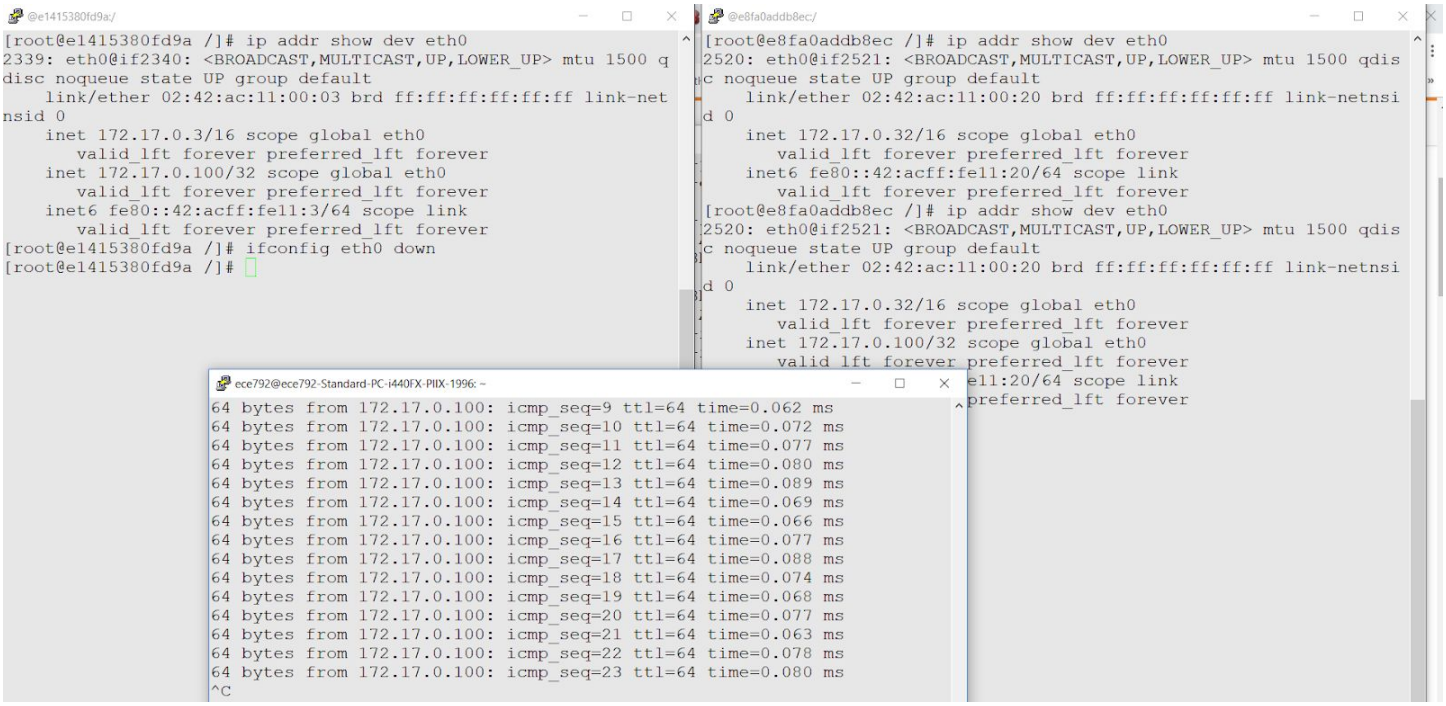
```
[root@d6297b417a48 /]# cat /named/var/forward.edu
$TTL 86400
@   IN  SOA    edu. root.edu. (
        2011071001  ;Serial
        10        ;Refresh
        5        ;Retry
        10      ;Expire
        20       ;Minimum TTL
)
@    IN  NS  t7.edu.
t7    IN  A   172.17.0.2
apidns.t7   IN  NS t7.edu.
api.apidns.t7           IN              NS              apidns.t7.edu.
api.t7      IN      CNAME         api.apidns.t7.edu.
webdns.t7   IN  NS t7.edu.
web.webdns.t7           IN              NS              webdns.t7.edu.
web.t7      IN      CNAME         web.webdns.t7.edu.
```

- Nslookup for t7.edu

```
[root@d6297b417a48 ~]# nslookup t7.edu
Server:         127.0.0.1
Address:        127.0.0.1#53

Name:   t7.edu
Address: 172.17.0.2
```

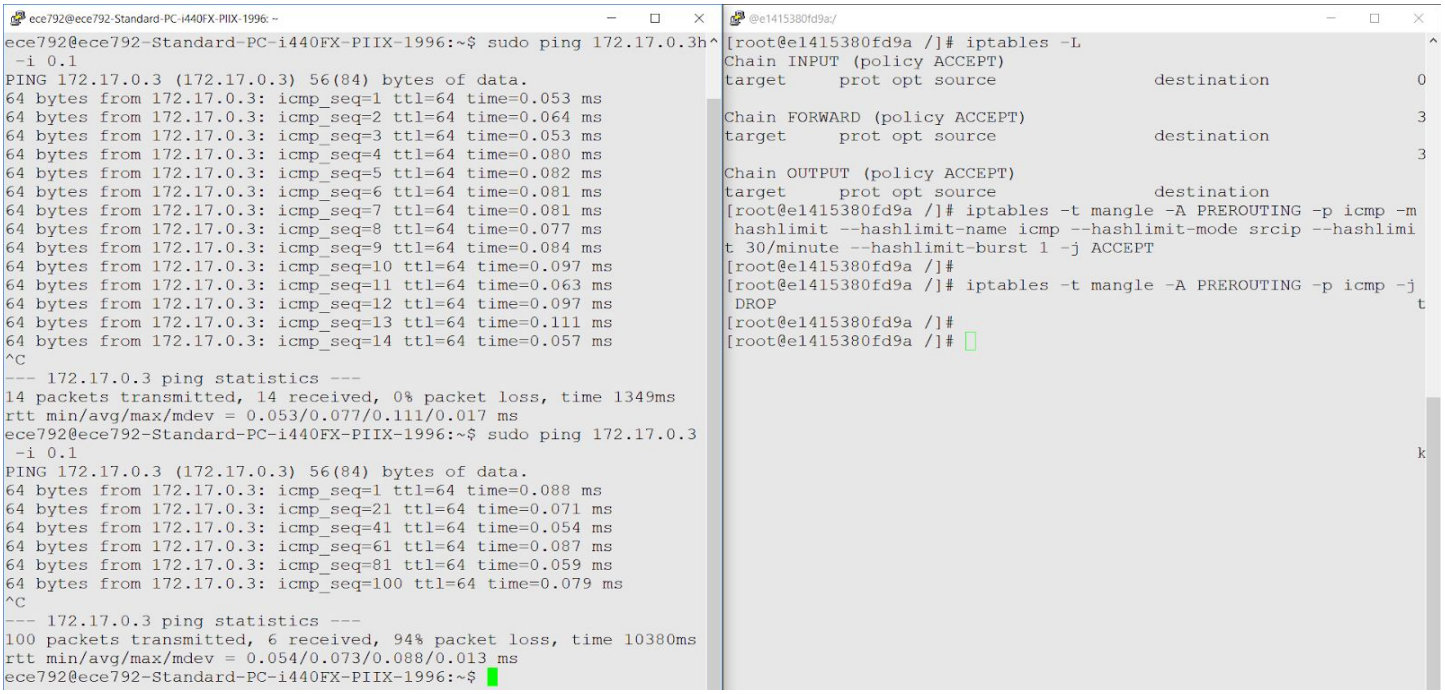
6.2.6.3 VRRP(High Availability)

The below screenshot of high availability using **keepalived** in containers. At the topmost lines, we can observe that both of them are in the 172.17.0.0/16 subnet. They both internally are configured to serve the 172.17.0.100 address. The ".3" has a priority of 100, whereas ".32" has a priority of 80. The script down_script.py turns an interface down for 30 seconds. As we can observe, when the script is executed, there is almost no loss of ping packets (in the bottom window). We can observe that the secondary i.e the node with lower priority picks up the IP as soon as it detects that the primary is down.



6.2.6.4 DOS Attack

The screenshot below is of the implementation on containers. After placing a rule which limits the traffic to 30 packets/minute, we can observe that the number of ping responses drops drastically. In the destination, we have added a rule to track and hash connections based on source IP. This can be extended to hosts internal to the VPC and external to the VPC. Different throughput limits can be set for trusted and untrusted hosts.



6.2.6.5 Deletion of containerized Infrastructure deployment

We have created python script self explainatory with user input command
Command: sudo python delete_containerized_infra.py

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo python delete_things.py
Enter your tenant ID
6
Enter (1) for VPC
(2) for subnet
(3) for container
1
Enter the vpc namet6vpc
Found  t6_mail  to be in vpc  t6vpc
Removing from config for subnet  t6_mail
Removing from config file for vpc  t6vpc
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
```

6.2.6.5 Controller SSH inside tenant DNS VMs

We have given a management feature for the tenant to login to any of its DNS containers inside its own vpc by logging into one controller container . Here, the tenant 1 has a controller container with ip address 38.0.0.139 and the tenant is logging into its DNS which has ip address 35.0.0.188

```
[root@83ea3ef2e884 /]# ifconfig t7test2vif1
t7test2vif1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 38.0.0.139  netmask 255.255.255.0  broadcast 38.0.0.255
        inet6 fe80::1858:ceff:fee9:9c81  prefixlen 64  scopeid 0x20<link>
        ether 1a:58:ce:e9:9c:81  txqueuelen 1000  (Ethernet)
        RX packets 181  bytes 21110 (20.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 142  bytes 15560 (15.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@83ea3ef2e884 /]# ssh root@35.0.0.188 -i id_rsa_private
System is booting up. See pam_nologin(8)
Last login: Sat Dec  8 03:28:55 2018 from v-guest-dhcp-dns-cool.d-cooldn-1.umnet.umich.edu
[root@4c799c5ce96c ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.27  netmask 255.255.0.0  broadcast 0.0.0.0
        inet6 fe80::42:acff:fe11:1b  prefixlen 64  scopeid 0x20<link>
        ether 02:42:ac:11:00:1b  txqueuelen 0  (Ethernet)
        RX packets 16655  bytes 911072 (889.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 63  bytes 5549 (5.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

t7web2vif1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 35.0.0.188  netmask 255.255.255.0  broadcast 35.0.0.255
        inet6 fe80::88a:4ff:fe51:76e  prefixlen 64  scopeid 0x20<link>
        ether 0a:8a:04:51:07:6e  txqueuelen 1000  (Ethernet)
        RX packets 324  bytes 31767 (31.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 189  bytes 24946 (24.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@4c799c5ce96c ~]#
```

7. References:

- https://sites.google.com/site/randomt3ch/home/how-to-add-new-dns-entry-in-docker-resolv-conf
- https://stackoverflow.com/questions/17157721/how-to-get-a-docker-containers-ip-address-from-the-host
- https://github.com/CentOS/CentOS-Dockerfiles/tree/master/bind/centos7
- https://medium.com/google-cloud/kubernetes-dns-proxy-with-services-d7d9e800c329
- https://medium.com/@kyle_26541/scale-kube-dns-3642bb7356c4
- https://github.com/kubernetes/kubernetes/blob/master/cluster/addons/dns/kube-dns/README.md
- https://medium.com/google-cloud/kubernetes-liveness-checks-4e73c631661f
- https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/
- https://github.com/kubernetes/dns/blob/master/docs/specification.md

8. Future Scope:

- Auto-Scaling can be added to the ges solution to take care of sudden surges in the DNS requests in case of special planned events
- We want to make this product “DNS-as-service” and code base as it could be deployed and used in AWS, GCP infrastructure (multi-cloud) and hybrid cloud infrastructure.
- The firewall can function better with a script which keeps track of previous IPs which have violated the throughput threshold and either blocking the users(external) or charging more to the client(internal guests).

9. Suggested Course name:

Networking Virtualization and DevOps