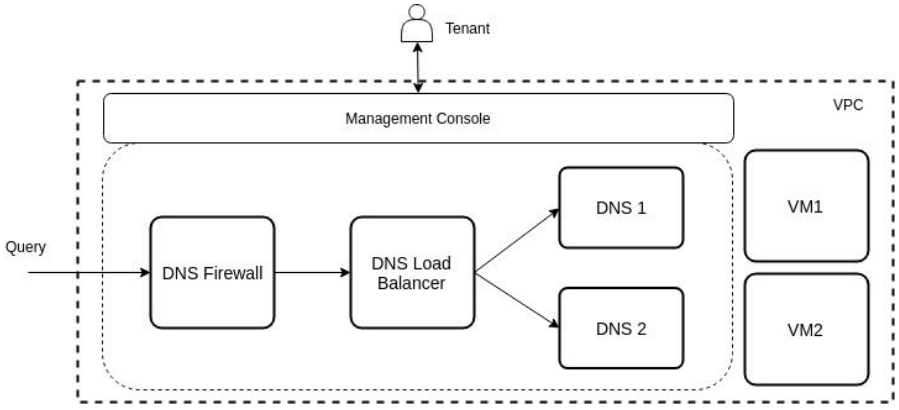
DNS-as-a-Service

Team 10

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

Introduction / Motivation:

- 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.
- 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
- 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:



Project Related Work:

- **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

Features Implementations Documentation:

The basic functionality of DNS-as-a-service is that it allows the tenant to create a private hosted zone. It is a container that holds tenant-specific DNS records information about how a tenant wants a DNS-as-a-Service to respond to DNS queries for a domain and its subdomains within one VPC that tenant has created.

I. Functional Features:

A. 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).
- The DNS server would monitor the health of all authoritative servers and do the load balancing accordingly.

B. 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 a public/private key pair of their own, which it uses to establish a symmetric key for inter DNS server communications (among peer DNS servers) and encrypt all traffic between them. If someone does hack 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.

II. Management Features:

A. User Interface / API:

1. 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.
2. REST API Endpoints:
    - 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.

B. 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.

References:

- DNS RFCs:
    - RFC 1034: https://tools.ietf.org/html/rfc1034
    - RFC 1035: https://tools.ietf.org/html/rfc1035

- Products:
    - Cloudflare DNS: https://www.cloudflare.com/dns/
    - Amazon Route 53: https://aws.amazon.com/route53/
    - NS1 DNS: https://ns1.com/products
    - Google Cloud DNS: https://cloud.google.com/dns/
    - Azure DNS: https://azure.microsoft.com/en-us/services/dns/
    - Rackspace Cloud DNS: https://www.rackspace.com/cloud/dns
    - https://www.siteground.com/kb/private_dns/

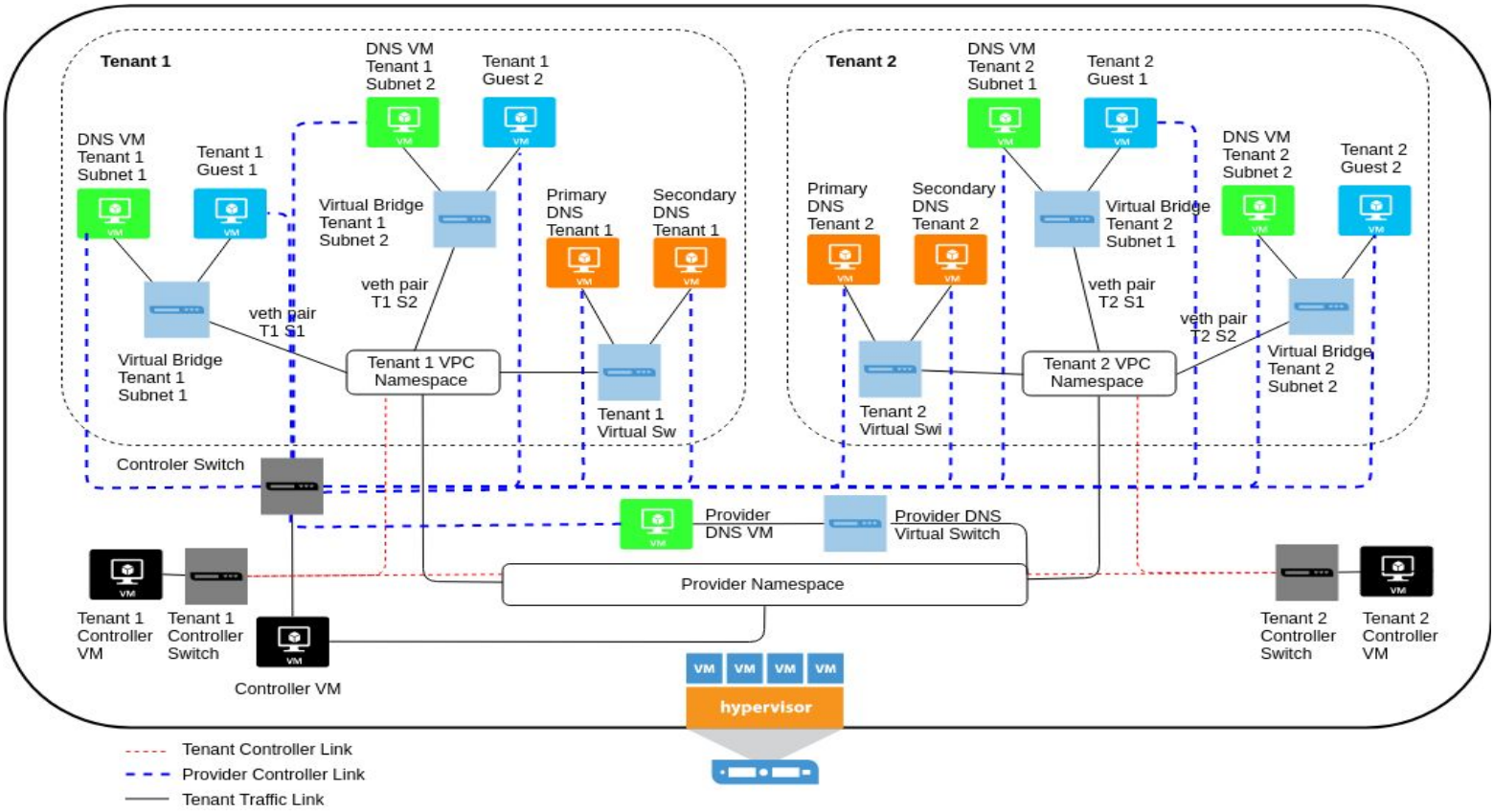
Milestone 1 feedback:

- Feature functions: -> ~~authoritative~~ (it is DNS servers which is part of management) -> backend servers (client's servers)
    - Content-Based Routing
    - Geolocation Based Routing
    - Server content / API end-point based routing

Milestone 2

Architecture

Following diagram represents the architecture that we have set-up. Dotted lines represent management links whereas

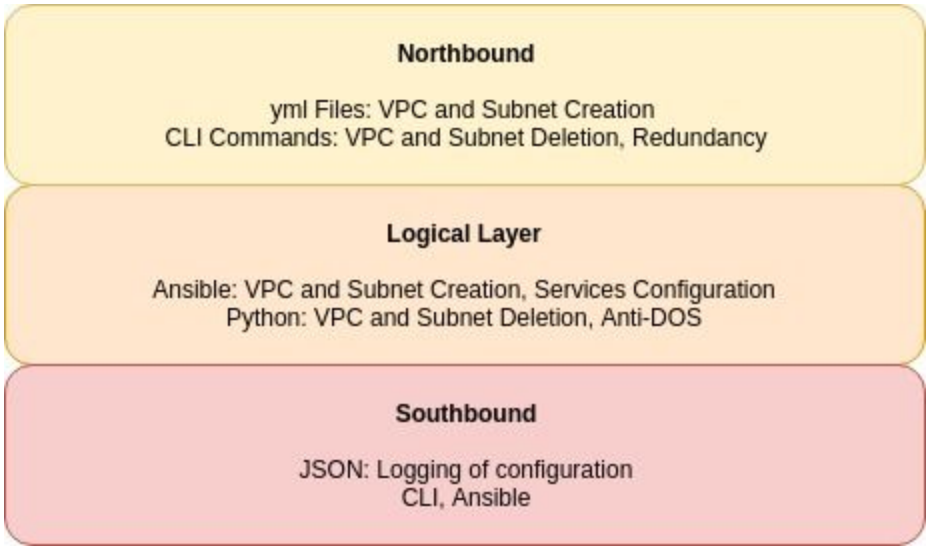


Role of each component:

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. Controller VM: Allows the provider to execute scripts to provision infrastructure for each tenant.

2. Tenant Infrastructure
    a. Tenant1 Guest1: VM which hosts the application used by the tenant.
    b. Primary and Secondary DNS servers: per tenant high available pair which are nameservers and maintain DNS entries for DNS VM in each subnet.
    c. 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.
    d. Tenant 1 controller VM: per tenant VM to allow clients to log into and manage their VMs

Functional Diagram



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.

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>; };
        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>

DNS Hierarchy

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.

Setting up high availability on VPC DNS:

1. Install the following on the servers on which HA needs to be configured.
    a. kernel-headers, kernel-devel, 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/second --hashlimit-burst 50 -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

API / CLI

For API/CLI code, instructions and description are included in the readme.
GitHub Repo: https://github.ncsu.edu/khchoksi/DNS-as-a-service

All the code and README is in *DNS-as-a-Service* folder.
The DNS configuration files for the TLD server, nameserver and authoritative server are attached with our submission in the *dnsconf* subfolder.

Reference Links:

- https://stackoverflow.com/questions/48784098/issue-with-kvm-libvirt-and-linux-namespaces
- https://ops.tips/blog/using-network-namespaces-and-bridge-to-isolate-servers/
- https://www.unixmen.com/setting-dns-server-centos-7/

  s3 inactive no yes

  Install DHCP server on Cent OS: https://www.tecmint.com/install-dhcp-server-in-centos-rhel-fedora/ (don't forget to provide default ip to eth0)
- https://backreference.org/2010/02/01/geolocation-aware-dns-with-bind/ (Geo-Location)
- http://www.zytrax.com/books/dns/ch8/ Great resource for bind9
- https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html#filters-for-formatting-data