Red Hat OpenStack Platform 12

DNS-as-a-Service Guide

Integrate DNS Management with Red Hat OpenStack Platform

Last Updated: 2018-03-02

Red Hat OpenStack Platform 12 DNS-as-a-Service Guide

Integrate DNS Management with Red Hat OpenStack Platform

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

A guide for integrating DNS with Red Hat OpenStack Platform.

Table of Contents

CHAPTER 1. OVERVIEW OF DNSAAS

Red Hat OpenStack Platform 12 includes a Technology Preview of DNS-as-a-Service (DNSaaS), also known as Designate. DNSaaS includes a REST API for domain and record management, is multi-tenanted, and integrates with OpenStack Identity Service (*keystone*) for authentication. DNSaaS includes a framework for integration with Compute (*nova*) and OpenStack Networking (*neutron*) notifications, allowing auto-generated DNS records. In addition, DNSaaS includes integration support for Bind9.

NOTE

DNS-as-a-Service (DNSaaS), also known as Designate, is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

1.1. TOPICS COVERED IN THIS GUIDE

- Manual DNSaaS installation steps, as DNSaaS is not currently included in Director deployment.
- Managing and configuring DNSaaS from the command line interface.
- Integration with Bind9, including auto-creation of instance records.

1.2. DNSAAS PREREQUISITES

- A fully functioning OpenStack Networking-based, non-HA OpenStack environment.
- An OpenStack Image Service (*glance*) image loaded, for testing auto-creation.

1.3. DNSAAS SERVICES

A deployment of DNSaaS includes the following components:

designate-api	Provides an OpenStack-native REST API.
designate-central	Handles requests and coordinates storage in the mysql database.
designate-mdns	A small MiniDNS server used only to communicate with other DNS servers over standard DNS protocol.
designate-pool-manager	Manages the states of the DNS servers that DNSaaS manages. Ensures the backend DNS servers are in sync with DNSaaS.
designate-sink	An optional service that is used to listen to nova and neutron notification events to trigger automatic record creation/deletion.

designate-agent	Used for DNS servers that cannot accept zone transfers (*AXFR*). Not needed for BIND backends.

NOTE

The **zone-manager** service is expected to be added in the next major release. It will run periodic tasks on zones to provide a mechanism for identifying lost events.

1.4. DNSAAS INTEGRATION WITH COMPUTE AND OPENSTACK NETWORKING

DNSaaS record management begins when the **`designate-sink`** service sends a message to **`designate-central`**, which then triggers the workflow described below:

1. **`designate-sink`** receives an *instance boot/delete* event from Compute, or a *floating IP add/remove* event from OpenStack Networking. These events are sent using the OpenStack message bus.

2. **`designate-sink`** constructs the FQDN of the host from the VM name and the configured domain ID (see below).

3. **`designate-sink`** tells **`designate-central`** to add/delete the record with the given name and IP address.

4. **`designate-central`** adds/deletes the record in the DNSaaS database (shared between **`designate-central`** and **`designate-mdns`**).

5. **`designate-central`** tells **`designate-pool-manager`** to send a **`DNS NOTIFY`** to the backend DNS server (BIND9) for this domain.

6. The backend DNS servers receive the **`DNS NOTIFY`** and send an **`AXFR`** (zone transfer) request to **`designate-mdns`**.

7. **`designate-mdns`** reads the changes from the database and sends them to the backend DNS servers in the **`AXFR`** response.

CHAPTER 2. MANUAL DNSAAS INSTALLATION

NOTE

Your server must be registered to receive the OpenStack packages. For more information, see https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/11/html-single/director_installation_and_usage/#sect-Registering_your_System

1. Install the DNSaaS packages on the *controller* node:

```
# yum install openstack-designate-api openstack-designate-central
openstack-designate-sink openstack-designate-pool-manager openstack-
designate-mdns openstack-designate-common python-designate python-
designateclient openstack-designate-agent
```

2. Create the DNSaaS and Pool Manager databases. Update the **`IDENTIFIED BY 'ComplexAlphanumericPassword'`** value to suit your environment.

```
# mysql -u root << EOF
CREATE DATABASE designate;
GRANT ALL ON designate.* TO 'designate'@'%' IDENTIFIED BY
'ComplexAlphanumericPassword';
GRANT ALL ON designate.* TO 'designate'@'localhost' IDENTIFIED BY
'ComplexAlphanumericPassword';
CREATE DATABASE designate_pool_manager;
GRANT ALL ON designate_pool_manager.* TO 'designate'@'%' IDENTIFIED BY
'ComplexAlphanumericPassword';
GRANT ALL ON designate_pool_manager.* TO 'designate'@'localhost'
IDENTIFIED BY 'ComplexAlphanumericPassword';
FLUSH PRIVILEGES;
quit
EOF
```

3. Create the DNSaaS service accounts and endpoint in OpenStack Identity (keystone): This example uses the DNSaaS host IP address **`192.168.100.20`**. You will likely need to update these steps to suit your environment.

```
$ openstack user create designate --password ComplexAlphanumericPassword -
-email designate@localhost
$ openstack role add --project service --user designate admin
$ openstack service create dns --name designate --description "Designate
DNS Service"
$ openstack endpoint create --region RegionOne --publicurl
http://192.168.100.20:9001 --internalurl http://192.168.100.20:9001 --
adminurl http://192.168.100.20:9001 designate
```

4. Add firewall rules for DNSaaS:

```
$ sudo iptables -I INPUT -p tcp -m multiport --dports 9001 -m comment --
comment "designate incoming" -j ACCEPT
$ sudo iptables -I INPUT -p tcp -m multiport --dports 5354 -m comment --
comment "Designate mdns incoming" -j ACCEPT
```

If hosting DNS locally, check that the required ports are open:

```
$ sudo iptables -I INPUT -p tcp -m multiport --dports 953 -m comment --
comment "rndc incoming - bind only" -j ACCEPT
$ sudo service iptables save; sudo service iptables restart
```

5. Configure the DNSaaS database connection: Be sure to enter your DNSaaS host IP address correctly in the steps below; replace **`ComplexAlphanumericPassword`** with the value that aligns with your environment.

```
$ crudini --set /etc/designate/designate.conf storage:sqlalchemy
connection
mysql://designate:ComplexAlphanumericPassword@192.168.100.20/designate
$ crudini --set /etc/designate/designate.conf storage:sqlalchemy
max_retries -1
$ crudini --set /etc/designate/designate.conf
pool_manager_cache:sqlalchemy connection
mysql://designate:ComplexAlphanumericPassword@192.168.100.20/designate_poo
l_manager
$ crudini --set /etc/designate/designate.conf
pool_manager_cache:sqlalchemy max_retries -1
```

6. Configure authentication to the Identity Service (*keystone*): Make certain that the **`admin_password`** option aligns with your environment.

```
$ crudini --set /etc/designate/designate.conf keystone_authtoken auth_uri
http://192.168.100.20:5000/v2.0
$ crudini --set /etc/designate/designate.conf keystone_authtoken
identity_uri http://192.168.100.20:35357/
$ crudini --set /etc/designate/designate.conf keystone_authtoken
admin_tenant_name service
$ crudini --set /etc/designate/designate.conf keystone_authtoken
admin_user designate
$ crudini --set /etc/designate/designate.conf keystone_authtoken
admin_password ComplexAlphanumericPassword
```

7. Configure the DNSaaS connection to RabbitMQ:

Make certain the **`rabbit_userid`** and **`rabbit_password`** options align with your environment.

```
$ crudini --set /etc/designate/designate.conf oslo_messaging_rabbit
rabbit_hosts 192.168.100.20:5672
$ crudini --set /etc/designate/designate.conf oslo_messaging_rabbit
rabbit_ha_queues False
$ crudini --set /etc/designate/designate.conf oslo_messaging_rabbit
rabbit_host 192.168.100.20
$ crudini --set /etc/designate/designate.conf oslo_messaging_rabbit
rabbit_port 5672
$ crudini --set /etc/designate/designate.conf oslo_messaging_rabbit
rabbit_userid amqp_user
$ crudini --set /etc/designate/designate.conf oslo_messaging_rabbit
rabbit_password ComplexAlphanumericPassword
$ crudini --set /etc/designate/designate.conf oslo_messaging_rabbit
rabbit_virtual_host /
```

8. Add the initial DNSaaS configuration:

```
$ crudini --set /etc/designate/designate.conf DEFAULT notification_driver
nova.openstack.common.notifier.rpc_notifier
$ crudini --set /etc/designate/designate.conf DEFAULT notification_driver
messaging
$ crudini --set /etc/designate/designate.conf DEFAULT notification_topics
notifications_designate
$ crudini --set /etc/designate/designate.conf service:api api_host 0.0.0.0
$ crudini --set /etc/designate/designate.conf service:api api_port 9001
$ crudini --set /etc/designate/designate.conf service:api auth_strategy
keystone
$ crudini --set /etc/designate/designate.conf service:api enable_api_v1
True
$ crudini --set /etc/designate/designate.conf service:api
enabled_extensions_v1 "diagnostics, quotas, reports, sync, touch"
$ crudini --set /etc/designate/designate.conf service:api enable_api_v2
True
$ crudini --set /etc/designate/designate.conf service:api
enabled_extensions_v2 "quotas, reports"
```

9. Configure the pool manager:

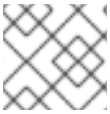
NOTE

At present, you will not yet configure a pool target as you have not selected a backend. That occurs later in this procedure.

The **pool_id** is hardcoded, so use the **UUID** shown below:

```
# pool_id=794ccc2c-d751-44fe-b57f-8894c9f5c842
# nameserver_id=$(uuidgen)
# target_id=$(uuidgen)
$ crudini --set /etc/designate/designate.conf service:pool_manager pool_id
$pool_id
$ crudini --set /etc/designate/designate.conf pool:$pool_id nameservers
$nameserver_id
$ crudini --set /etc/designate/designate.conf pool:$pool_id targets
$target_id
$ crudini --set /etc/designate/designate.conf
pool_nameserver:$nameserver_id port 53
$ crudini --set /etc/designate/designate.conf
pool_nameserver:$nameserver_id host 192.168.100.20
```

10. Configure the DNSaaS Sink:



NOTE

For now, you will not configure the domain used by sink (as it does not exist yet).

```
$ crudini --set /etc/designate/designate.conf service:sink
enabled_notification_handlers "nova_fixed, neutron_floatingip"
$ crudini --set /etc/designate/designate.conf handler:nova_fixed
```

```
notification_topics notifications_designate
$ crudini --set /etc/designate/designate.conf handler:nova_fixed
control_exchange nova
$ crudini --set /etc/designate/designate.conf handler:nova_fixed format "%
(display_name)s.%(domain)s"
$ crudini --set /etc/designate/designate.conf handler:neutron_floatingip
notification_topics notifications_designate
$ crudini --set /etc/designate/designate.conf handler:neutron_floatingip
control_exchange neutron
$ crudini --set /etc/designate/designate.conf handler:neutron_floatingip
format "%(octet0)s-%(octet1)s-%(octet2)s-%(octet3)s.%(domain)s"
```

11. Configure Compute and OpenStack Networking to send notifications

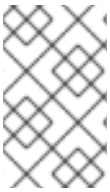
NOTE

Ceilometer's agent also listens and consumes notifications. Create a specific **`Designate`** notifications queue (as shown below) so they don't conflict.

OpenStack Compute in the *Kilo* release switched to **`messaging`** as its notification driver; previously it was **`nova.openstack.common.notifier.rpc_notifier`**

```
$ crudini --set /etc/nova/nova.conf DEFAULT notification_topics
notifications,notifications_designate
$ crudini --set /etc/nova/nova.conf DEFAULT notify_on_state_change
vm_and_task_state
$ crudini --set /etc/nova/nova.conf DEFAULT instance_usage_audit_period
hour
$ crudini --set /etc/nova/nova.conf DEFAULT instance_usage_audit true
$ crudini --set /etc/neutron/neutron.conf DEFAULT notification_driver
neutron.openstack.common.notifier.rpc_notifier
$ crudini --set /etc/neutron/neutron.conf DEFAULT notification_topics
notifications,notifications_designate
$ sudo systemctl restart nova.service
$ sudo systemctl restart neutron.service
```

12. Manually verify the **`notification_driver`** in *nova.conf*:



NOTE

Due to the possibility of multiple **`notification_drivers`** in *nova.conf*, the *crudini* command might cause problems. Check in the **`DEFAULT`** section to ensure you have these two entries:

```
notification_driver=ceilometer.compute.nova_notifier
notification_driver=messaging
```

NOTE

If using a separate Compute node, it will need the following settings in *nova.conf*:

```
notification_driver
=nova.openstack.common.notifier.rabbit_notifier,ceilometer.comp
ute.nova_notifier
notification_driver =messaging
notification_topics=notifications,notifications_designate
```

13. Sync the DNSaaS and Pool Manager cache:

```
# designate-manage database sync
# designate-manage pool-manager-cache sync
```

14. Enable and start the DNSaaS services:

```
# systemctl enable designate-central
# systemctl enable designate-api
# systemctl enable designate-mdns
# systemctl enable designate-pool-manager
# systemctl start designate-central
# systemctl start designate-api
# systemctl start designate-mdns
# systemctl start designate-pool-manager
```

NOTE

At this point you have not created a DNS target for your pool, so don’t expect a functioning DNSaaS deployment yet.

CHAPTER 3. INSTALL AND CONFIGURE BIND9

These steps install Bind9, and then configure integration with DNSaaS.

3.1. BASIC BIND INSTALLATION

1. Install the *BIND* packages:

```
# yum install bind bind-utils
```

2. Configure *named* to listen for incoming connections:

```
# cp /etc/named.conf /etc/named.conf.orig
# sed -i -e "s/listen-on port.*/listen-on port 53 { 127.0.0.1;
192.168.100.20; };/" /etc/named.conf
```

3.2. CONFIGURE BIND

1. Write to */etc/rndc.key*:

```
# rndc-confgen -a
```

2. Add the following before **`options`**

```
# sed -i '/^options.*/i \
include "/etc/rndc.key"; \
controls { \
        inet 127.0.0.1 allow { localhost; } keys { "rndc-key"; }; \
};' /etc/named.conf
```

3. Remove a few existing options you will rewrite later:

```
# sed -i '/allow-query.*/d' /etc/named.conf
# sed -i '/recursion.*/d' /etc/named.conf
```

4. Add the following after **`options`**:

```
# sed -i '/^options.*/a \
        allow-new-zones yes; \
        allow-query { any; }; \
        recursion no;' /etc/named.conf
```

5. Create the *rndc* configuration. For the Compute node, the rndc configuration must point to the DNS server. For example:

```
# cat << EOF > /etc/rndc.conf
include "/etc/rndc.key";
options {
        default-key "rndc-key";
        default-server 192.168.100.20;
        default-port 953;
};
```

```
EOF
```

6. Review the *named* configuration:

```
# named-checkconf /etc/named.conf
```

7. Correct the file permissions:

```
# setsebool -P named_write_master_zones on
# chmod g+w /var/named
# chown named:named /etc/rndc.conf
# chown named:named /etc/rndc.key
# chmod 600 /etc/rndc.key
```

8. Enable and start the *named* service:

```
# systemctl enable named
# systemctl start named
```

9. Validate *named* and *rndc*:

```
# dig @localhost localhost
# rndc status
```

3.3. CONFIGURE THE DNSAAS POOL TARGET FOR BIND

1. Set the pool target configuration:

```
$ crudini --set /etc/designate/designate.conf pool_target:$target_id type
bind9
$ crudini --set /etc/designate/designate.conf pool_target:$target_id
options "rndc_host: 192.168.100.20, rndc_port: 953, rndc_config_file:
/etc/rndc.conf, rndc_key_file: /etc/rndc.key"
$ crudini --set /etc/designate/designate.conf pool_target:$target_id
masters 192.168.100.20:5354
```

2. Restart DNSaaS to apply your pool changes:

```
# systemctl restart designate-api
# systemctl restart designate-central
# systemctl restart designate-mdns
# systemctl restart designate-pool-manager
# systemctl restart designate-sink
```

3.4. TEST BIND

1. Perform the diagnostic commands below:

```
# netstat -tap | grep named
# netstat -tulpn | grep 53
# dig @192.168.100.20
```

2. Check the DNSaaS Logs for errors. Ignore errors in Sink for now, as you have not modified its configuration.

```
# cd /var/log/designate
# tail api.log
# tail central.log
# tail mdns.log
# tail pool-manager.log
# tail sink.log
```

3.5. TEST DNSAAS INTEGRATION WITH BIND9

1. Create an entry for your server:

```
# designate server-create --name $(hostname).
```

2. Verify your DNS server record was previously created:

```
# designate server-list
```

3. Create a domain (don't forget the **.** at the end of the **--name** option)

```
# designate domain-list
# designate domain-create --name example.com. --email root@example.com
# DOMAINID=$(designate domain-list | grep example.com | awk '{print $2}')
```

NOTE

When creating a domain from designate against BIND, it is basically running a command similiar to this:

```
# rndc -s 192.168.122.41 -p 953 -c /etc/rndc.conf -k /etc/rndc.key addzone
example.com '{ type slave; masters { 192.168.122.41 port 5354; }; file
"slave.example.com.ff532e15-55a9-4966-8f1e-b3eddb2891ba"; };'
```

4. Create a record and test lookup (don't forget the **.** at the end of the **--name** option)

```
# designate record-create --name server1.example.com. --type A --data
1.2.3.4 $DOMAINID
# dig +short -p 53 @192.168.100.20 server1.example.com A
```

3.6. CONFIGURE AUTO-GENERATION OF DNS RECORDS (NOVA FIXED AND NEUTRON FLOATING)

1. Modify the DNSaaS configuration for the example domain:

```
$ crudini --set /etc/designate/designate.conf handler:nova_fixed domain_id
$DOMAINID
$ crudini --set /etc/designate/designate.conf handler:neutron_floatingip
domain_id $DOMAINID
```

```
# systemctl restart designate-api
# systemctl restart designate-central
# systemctl restart designate-mdns
# systemctl restart designate-pool-manager
# systemctl restart designate-sink
```

2. Test OpenStack Compute (nova) record creation:

```
# glance image-list
# neutron net-list
# nova boot testserver --flavor m1.tiny --image cirros-0.3.4-x86_64 --key-
name yourkey --security-groups default --nic net-id=<Private Net ID>
```

3. Check the Sink log:

Once the instance is up, you should see a **`create_record`** entry, if it has picked up the notification correctly:

```
# tail /var/log/designate/sink.log
```

Check in BIND

```
# dig +short @192.168.100.20 testserver.example.com
```

If this doesn't work, you can also check the files in **`/var/named`**.

3.7. TEST OPENSTACK NETWORKING FLOATING IP RECORD CREATION

1. Perform the diagnostic commands below (replace **`pubnet1`** with a name appropriate for your environment):

```
# FLOATINGIP=$(neutron floatingip-create pubnet1 | grep
floating_ip_address | awk '{print $4}')
# nova add-floating-ip testserver $FLOATINGIP
# DNSRESULT=$(echo $FLOATINGIP |sed 's/\./-/g').example.com
# dig +short @192.168.100.20 $DNSRESULT
```

2. You should see a **`create_record`** event in the log file:

```
# tail /var/log/designate/sink.log
```

3.8. CLEANUP OPENSTACK NETWORKING AND COMPUTE DNS ENTRIES

1. Remove the test floating IP created previously:

```
# nova remove-floating-ip testserver $FLOATINGIP
```

2. You should see a **`delete_record`** event in the log file:

```
# tail /var/log/designate/sink.log
```

And the record should now be removed.

3. Remove the *testserver* created previously:

```
# designate record-list $DOMAINID
# nova delete testserver
```

You should see another **`delete_record`** entry in the log file:

```
# tail /var/log/designate/sink.log
```