Branch: **master** ▾ Find file Copy path

openshift-toolbox / ocp4_upi / docs / **0.prereqs.md**

 **christianh814** Update 0.prereqs.md

`123eb92` on Oct 17, 2019

1 contributor

Raw Blame History

`552 lines (439 sloc)` `14.6 KB`

Prereqs

This is probably the longest and most important part of this guide. The prereqs are "heavier" in this release. So I'll break it down into sections

- Infra
- DHCP
- DNS
- Load Balancer
- WebServer

Remember, a lot of this can be done for you if you use my helper node playbook.

Infra

This is a minimum setup

Create 6 "blank" VMs (or use BareMetal) for OpenShift itself (don't install an OS on this just yet)

- 3 Masters
  - 4 vCPUs
  - 16 GB RAM
  - 120 GB HD
- 2 Workers (I did 3; but 2 works)
  - 2 vCPUs
  - 8 GB RAM
  - 120 GB HD
- 1 Boostrap (you remove this after the install is done)
  - 4 vCPUs
  - 16 GB RAM

- 120 GB HD

Create 1 VM for the LoadBalancer (I used CentOS 7 but any Linux will do)

- LoadBalancer VM
  - 2 vCPUs
  - 4 GB RAM
  - 30 GB HD

Create 1 VM for DNS (You can use Centos/Fedora/RHEL)

- DNS VM
  - 2 vCPUs (if using IDM use 4 vCPUs)
  - 4 GB RAM
  - 30 GB HD (if Using IDM set this to 50GB HD)
  - You can also use this for your Webserver too
  - You can install the DHCP components here too

If you don't have a lot of resources you can combine the LB and DNS server but I don't recommend it. If you don't have enough resources buy another server/box. See the min requirements for OpenShift 4...it's a BEAST!

DHCP

Right now, with 4.2, you can't set static ip addresses with the installer. So "static" DHCP (where you reserve the MAC address to a specific IP address in your network) is the way to go.

Here is a high level overview on how to do it...**I actually did mine on my router**; which is the way I recommend that you do it...but if you don't; here is how to install a simple DHCP server.

Install packages

```
yum -y install dhcp
```

Next set it up to where it'll start on boot

```
systemctl enable dhcpd
```

Make note of the MAC address of you LB and OpenShift infra. Use this to create `/etc/dhcp/dhcpd.conf`. The below is an EXAMPLE. Use it to create your own...

```
authoritative;
ddns-update-style interim;
default-lease-time 14400;
max-lease-time 14400;

        option routers                  192.168.1.1;
        option broadcast-address        192.168.1.255;
        option subnet-mask              255.255.255.0;
        option domain-name-servers      192.168.1.2, 192.168.1.3;
        option domain-name              "example.com";

        subnet 192.168.1.0 netmask 255.255.255.0 {
        pool {
                range 192.168.1.100 192.168.1.200;
                # Static entries
                host bootstrap { hardware ethernet 08:ed:b9:2d:82:4a; fixed-address
```

```
192.168.1.104; }
                host lb { hardware ethernet 08:ed:b9:2d:82:4b; fixed-address
192.168.1.105; }
                host master0 { hardware ethernet 01:1d:b9:2d:82:4c; fixed-address
192.168.1.100; }
                host master1 { hardware ethernet 02:ad:b9:2d:82:4d; fixed-address
192.168.1.101; }
                host master2 { hardware ethernet 03:4d:b9:2d:82:4e; fixed-address
192.168.1.102; }
                host worker0 { hardware ethernet 04:7d:b9:2d:82:4f; fixed-address
192.168.1.110; }
                host worker1 { hardware ethernet 05:bd:b9:2d:82:5e; fixed-address
192.168.1.111; }
                host worker2 { hardware ethernet 08:8d:b9:2d:82:1b; fixed-address
192.168.1.112; }
                # this will not give out addresses to hosts not listed above
                deny unknown-clients;
        }
}
```

Configure firewall stuff

```
firewall-cmd --add-service=dhcp --permanent
firewall-cmd --reload
```

Now start the service

```
systemctl start dhcpd
```

^ Again....**I DON'T RECOMMEND THIS** ...if you can do this in your router...do it there (that's what I did).

DNS

OCP4 uses SRV recrods for autodiscovery. More information can be found here. I used idm (a.k.a FreeIPA) for DNS. But that's a little heavy (it also installs Kerberos, LDAP, Dogtag, etc).

So if you JUST need DNS; here's a quick overview on how to set it up. (the below assumes Fedora but can work on RHEL/CentOS)

Install it

```
dnf -y install bind bind-utils
```

Edit the `/etc/named.conf` file to look similar to what's below. I made only 3 changes

- `listen-on` was changed to read `any`
- `allow-query` was changed to read `any`
- `forward only;` I added this for DNS to foward queries
- `forwarders { 8.8.8.8; 8.8.4.4; };` forwarding those quieres to Google's DNS servers
- The entries for my zonefiles (noted by comments)

THE DEFAULT FILE `/etc/named.conf` is fine ...just modify/add the things I mentioned above

```
//
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
```

```
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//

options {
        listen-on port 53 { any; };
        listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        allow-query     { any; };

        /*
         - If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
         - If you are building a RECURSIVE (caching) DNS server, you need to enable
           recursion.
         - If your recursive DNS server has a public IP address, you MUST enable access
           control to limit queries to your legitimate users. Failing to do so will
           cause your server to become part of large scale DNS amplification
           attacks. Implementing BCP38 within your network would greatly
           reduce such attack surface
        */
        recursion yes;

        /* Fowarders */
        forward only;
        forwarders { 8.8.8.8; 8.8.4.4; };

        dnssec-enable yes;
        dnssec-validation yes;

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

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

        /* https://fedoraproject.org/wiki/Changes/CryptoPolicy */
        include "/etc/crypto-policies/back-ends/bind.config";
};

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

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

############ Add what's between these comments ############
zone "example.com" IN {
        type master;
        file    "example.com";
};

zone "1.168.192.in-addr.arpa" IN {
        type    master;
        file    "192.168.1.db";
};
############################################################

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

Create `/var/named/example.com` zone file with the following entries. Making sure you've set your SRV records. NOTE, whatever your subdomain is (in this case it's `ocp4`) **WILL BE** the name of your cluster when you install.

```
$TTL 1W
@       IN      SOA     ns1.example.com.        root (
                        2019052300      ; serial
                        3H              ; refresh (3 hours)
                        30M             ; retry (30 minutes)
                        2W              ; expiry (2 weeks)
                        1W )            ; minimum (1 week)
        IN      NS      ns1.example.com.
        IN      MX 10   smtp.example.com.
;
;
ns1     IN      A       192.168.1.2
smtp    IN      A       192.168.1.2
;
; The api points to the IP of your load balancer
api.ocp4        IN      A       192.168.1.105
api-int.ocp4    IN      A       192.168.1.105
;
; The wildcard also points to the load balancer
*.apps.ocp4     IN      A       192.168.1.105
;
; Create entry for the bootstrap host
bootstrap.ocp4  IN      A       192.168.1.104
;
; Create entries for the master hosts
master0.ocp4    IN      A       192.168.1.100
master1.ocp4    IN      A       192.168.1.101
master2.ocp4    IN      A       192.168.1.102
;
; Create entries for the worker hosts
worker0.ocp4    IN      A       192.168.1.110
worker1.ocp4    IN      A       192.168.1.111
worker2.ocp4    IN      A       192.168.1.112
;
; The ETCd cluster lives on the masters...so point these to the IP of the masters
etcd-0.ocp4     IN      A       192.168.1.100
etcd-1.ocp4     IN      A       192.168.1.101
etcd-2.ocp4     IN      A       192.168.1.102
;
; The SRV records are IMPORTANT....make sure you get these right...note the trailing dot
at the end...
_etcd-server-ssl._tcp.ocp4      IN      SRV     0 10 2380 etcd-0.ocp4.example.com.
_etcd-server-ssl._tcp.ocp4      IN      SRV     0 10 2380 etcd-1.ocp4.example.com.
_etcd-server-ssl._tcp.ocp4      IN      SRV     0 10 2380 etcd-2.ocp4.example.com.
;
;EOF
```

Alright, now create the `/var/named/192.168.1.db` file for reverse DNS. This is important because this is how the host sets it's hostname on boot.

```
$TTL 1W
@       IN      SOA     ns1.example.com.        root (
                        2019052300      ; serial
                        3H              ; refresh (3 hours)
                        30M             ; retry (30 minutes)
                        2W              ; expiry (2 weeks)
                        1W )            ; minimum (1 week)
        IN      NS      ns1.example.com.
;
; syntax is "last octet" and the host must have fqdn with trailing dot
100     IN      PTR     master0.ocp4.example.com.
```

```
101     IN      PTR     master1.ocp4.example.com.
102     IN      PTR     master2.ocp4.example.com.
;
104     IN      PTR     bootstrap.ocp4.example.com.
;
105     IN      PTR     api.ocp4.example.com.
105     IN      PTR     api-int.ocp4.example.com.
;
110     IN      PTR     worker0.ocp4.example.com.
111     IN      PTR     worker1.ocp4.example.com.
112     IN      PTR     worker2.ocp4.example.com.
;
;
;EOF
```

Make sure SELinux is happy

```
restorecon -vR /var/named
```

If you're using a firewall; open up those ports

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

Enable/start it

```
systemctl enable --now named
```

Make sure it's up

```
systemctl status named
```

LB

I used HAProxy on CentOS 7 ...it's pretty straight forward.

Install it on your lb (192.168.1.105) server

```
yum -y install haproxy
```

Edit the `/etc/haproxy/haproxy.cfg` file to look like this...

```
#---------------------------------------------------------------------
# Example configuration for a possible web application.  See the
# full configuration options online.
#
#   http://haproxy.1wt.eu/download/1.4/doc/configuration.txt
#
#---------------------------------------------------------------------

#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
    # to have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
```

```
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

#---------------------------------------------------------------------

listen stats
    bind :9000
    mode http
    stats enable
    stats uri /
    monitor-uri /healthz


frontend openshift-api-server
    bind *:6443
    default_backend openshift-api-server
    mode tcp
    option tcplog

backend openshift-api-server
    balance source
    mode tcp
    server boot 192.168.1.104:6443 check
    server master-0 192.168.1.100:6443 check
    server master-1 192.168.1.101:6443 check
    server master-2 192.168.1.102:6443 check
```

```
frontend machine-config-server
    bind *:22623
    default_backend machine-config-server
    mode tcp
    option tcplog

backend machine-config-server
    balance source
    mode tcp
    server boot 192.168.1.104:22623 check
    server master-0 192.168.1.100:22623 check
    server master-1 192.168.1.101:22623 check
    server master-2 192.168.1.102:22623 check

frontend ingress-http
    bind *:80
    default_backend ingress-http
    mode tcp
    option tcplog

backend ingress-http
    balance source
    mode tcp
    server worker-0 192.168.1.110:80 check
    server worker-1 192.168.1.111:80 check
    server worker-2 192.168.1.112:80 check

frontend ingress-https
    bind *:443
    default_backend ingress-https
    mode tcp
    option tcplog

backend ingress-https
    balance source
    mode tcp
    server worker-0 192.168.1.110:443 check
    server worker-1 192.168.1.111:443 check
    server worker-2 192.168.1.112:443 check

#---------------------------------------------------------------------
```

Set up the firewall

```
firewall-cmd --permanent --add-port=9000/tcp \
--add-port=443/tcp \
--add-port=80/tcp \
--add-port=6443/tcp \
--add-port=6443/udp \
--add-port=22623/tcp \
--add-port=22623/udp
firewall-cmd --reload
```

Make SELinux Happy

```
setsebool -P haproxy_connect_any on
```

Start/enable HAProxy

```
systemctl enable haproxy --now
```

Verify

```
systemctl status haproxy
```

Visit the status page

```
firefox http://192.168.1.105:9000
```

^ This page is important to keep up when installing

Webserver

You need a place to store the bios file and ignition files. You can create a webserver for this (make sure you change `/etc/httpd/conf/httpd.conf` file to "Listen" on port `8080` since the HAProxy LB is listening on `80`)

```
yum -y install httpd
systemctl enable --now httpd
firewall-cmd --add-service=8080/tcp --permanent
firewall-cmd --reload
```

^ I didn't do this...since I'm on a linux laptop I just ran the following from my laptop

```
python3 -m http.server -d /path/to/install/dir 8080
```

But the apache webserver is fine too

Conclusion

Do the following commands to check everything

DNS

etcd SRV recrods

```
$ dig _etcd-server-ssl._tcp.ocp4.example.com SRV +short
0 10 2380 etcd-0.ocp4.example.com.
0 10 2380 etcd-2.ocp4.example.com.
0 10 2380 etcd-1.ocp4.example.com.
```

Also check your LB

```
$ dig api.ocp4.example.com +short
192.168.1.105
$ dig api-int.ocp4.example.com +short
192.168.1.105
```

You should check the hosts names too of your nodes

LB

Simple curl will return the health of the LB itself

```
$ curl api.ocp4.example.com:9000/healthz
<html><body><h1>200 OK</h1>
```

```
Service ready.
</body></html>
```

return to the index page