OpenShift Cluster Setup Guide

This document is structured into multiple parts required to create an OpenShift cluster in an isolated or disconnected environment:

- **Part 1**: Setting up the Router (DHCP + DNS using `dnsmasq` and `BIND`)
- **Part 2**: Creating the OpenShift Master Node
- **Part 3**: Creating the OpenShift Worker Nodes

Part 1: OpenShift Prerequisites — DHCP & DNS Router Setup with `dnsmasq` and `BIND`

Why This Setup Is Needed

When building an OpenShift cluster in a disconnected or isolated environment (e.g., lab, testbed, or private cloud), a local DHCP and DNS system is essential to:

- Assign static and dynamic IPs to cluster nodes.
- Resolve cluster-internal domains like `api.<cluster>.<domain>`.
- Forward general DNS traffic for internet access (e.g., yum, updates).

A Fedora/RHEL laptop or system can act as a router to provide both internet access and domain services to OpenShift nodes connected via a switch.

Global Prerequisites

- Fedora or RHEL-based system with `sudo` access
- Active Wi-Fi connection for internet
- Ethernet port (e.g., `enp2s0`) wired to a switch
- At least one other device (worker node or client) connected to the switch
- NetworkManager must be used to manage interfaces

Step-by-Step Configuration

Step 1: Identify Interfaces

```
ip a
```

Explanation:

- `ip`: tool to show/manipulate IP and routing
- `a` (short for `addr`): show all IP addresses

```
ip link show
```

Explanation:

- `ip link`: show all network interfaces and their status

Step 2: Create Ethernet Connection Profile

```
sudo nmcli con add type ethernet con-name ssm-router ifname enp2s0 ipv4.method manual ipv4.addresses 192.168.10.1/24
```

Explanation:

- `sudo`: run with root privileges
- `nmcli con add`: add a new network connection
- `type ethernet`: specify it's a wired connection
- `con-name ssm-router`: name the connection
- `ifname enp2s0`: assign it to Ethernet interface `enp2s0`
- `ipv4.method manual`: use a static IP
- `ipv4.addresses`: assign IP `192.168.10.1` with subnet /24

Step 3: Activate the New Profile

```
sudo nmcli con down enp2s0
sudo nmcli con up enp2s0
ip a show enp2s0
```

Explanation:

- Deactivates any existing config, activates the new one, and checks the IP

Step 4: Enable IP Forwarding

```
sudo nano /etc/sysctl.conf
```

Add this line at the end:

```
net.ipv4.ip_forward = 1
```

Then apply changes:

```
sudo sysctl -p
```

Explanation:

- Enables packet forwarding between interfaces for routing

Step 5: Configure NAT (Masquerading)

```
sudo dnf install iptables iptables-services -y
```

Explanation:

- Installs NAT firewall tools

```
sudo iptables -t nat -A POSTROUTING -o wlp3s0 -j MASQUERADE
```

Explanation:

- Add a NAT rule to allow LAN traffic to access internet via Wi-Fi (`wlp3s0`)

```
sudo iptables -A FORWARD -i enp2s0 -j ACCEPT
```

Explanation:

- Allow traffic from Ethernet interface to be forwarded

```
sudo service iptables save
sudo systemctl enable iptables
```

Explanation:

- Save firewall config and enable on boot

Step 6: Install & Configure dnsmasq (DHCP)

```
sudo dnf install dnsmasq -y
sudo vi /etc/dnsmasq.conf
```

Paste the following config (with comments):

```
port=0  # Disable DNS functionality in dnsmasq
interface=enp2s0  # Bind to Ethernet only
bind-interfaces  # Bind only to the specified interface
dhcp-range=192.168.10.10,192.168.10.150,24h  # IP pool

# Static mappings
dhcp-host=3c:2c:30:f1:d1:2d,openshift-master,192.168.10.11,infinite
dhcp-host=3c:2c:30:f1:d7:c1,openshift-worker1,192.168.10.12,infinite

# Set default gateway and DNS
dhcp-option=3,192.168.10.1
dhcp-option=6,192.168.10.1
dhcp-authoritative  # Force this DHCP server to act as the authority
```

```
sudo systemctl enable --now dnsmasq
```

Explanation:

- Enable and start the DHCP server

Step 7: Install & Configure BIND (DNS)

```
sudo dnf install bind bind-utils -y
sudo vi /etc/named.conf
```

Paste this config:

```
options {
    directory "/var/named";
    listen-on port 53 { 127.0.0.1; 192.168.10.1; };
    allow-query { any; };
    recursion yes;
    forwarders { 8.8.8.8; 1.1.1.1; };
    dnssec-enable yes;
    dnssec-validation yes;
};

zone "ssm.dsw.ost" IN {
    type master;
    file "/var/named/ssm.dsw.ost.zone";
    allow-update { none; };
};
```

Then create zone file:

```
sudo vi /var/named/ssm.dsw.ost.zone
```

Paste:

```
$TTL 86400
@   IN  SOA     ns1.ssm.dsw.ost. admin.ssm.dsw.ost. (
                2025060401 ; Serial
                3600       ; Refresh
                1800       ; Retry
                604800     ; Expire
                86400 )    ; Minimum TTL

    IN  NS      ns1.ssm.dsw.ost.
ns1.ssm.dsw.ost.                IN A 192.168.10.1
openshift-master.ssm.dsw.ost.   IN A 192.168.10.131
openshift-worker.ssm.dsw.ost.   IN A 192.168.10.11
api.ssm.dsw.ost.                IN A 192.168.10.131
api-int.ssm.dsw.ost.            IN A 192.168.10.131
*.apps.ssm.dsw.ost.             IN A 192.168.10.131
```

Validate and reload:

```
sudo named-checkzone ssm.dsw.ost /var/named/ssm.dsw.ost.zone
sudo rndc reload ssm.dsw.ost
```

Or reload whole service:

```
sudo systemctl reload named
```

Step 8: Enable DNS and DHCP Services

```
sudo systemctl enable named
sudo systemctl start named
sudo systemctl enable dnsmasq
sudo systemctl start dnsmasq
```

Explanation:

- Make sure services are running and persistent after reboot

Step 9: Configure Firewalld (if enabled)

```
sudo firewall-cmd --add-masquerade --permanent
sudo firewall-cmd --zone=internal --add-interface=enp2s0 --permanent
sudo firewall-cmd --zone=internal --change-interface=enp2s0
sudo firewall-cmd --zone=internal --add-service=dns --add-service=dhcp --permanent
sudo firewall-cmd --reload
```

Explanation:

- Open firewall rules and reload the configuration

Client Testing

```
ip a
```

Check if client gets IP in `192.168.10.x` range

```
ping 192.168.10.1
```

Test router reachability

```
ping 8.8.8.8
```

Test internet access via IP

```
ping google.com
```

Test internet access with DNS

```
nslookup google.com
```

Test DNS server resolution

```
nslookup api.ssm.dsw.ost 192.168.10.1
nslookup openshift-master.ssm.dsw.ost 192.168.10.1
nslookup openshift-worker.ssm.dsw.ost 192.168.10.1
nslookup api-int.ssm.dsw.ost 192.168.10.1
nslookup apps.ssm.dsw.ost 192.168.10.1
```

Test internal domain resolution

Troubleshooting (Common Issues & Fixes)

Issue 1: No DHCP Lease Assigned to Clients

Symptoms:

- Clients do not get IP addresses
- No DHCPDISCOVER or DHCPOFFER traffic observed

Causes & Fixes:

- `dnsmasq` is not bound to the correct interface
  Fix: Ensure the following lines are in `/etc/dnsmasq.conf`:

  ```
  interface=enp2s0
  bind-interfaces
  ```

- The internal Ethernet interface (`enp2s0`) doesn't have a static IP
  Fix: Assign a static IP:

  ```
  sudo nmcli con add type ethernet con-name ssm-router ifname enp2s0 ipv4.method manual ipv4.addresses 192.168.10.1/24
  ```

Issue 2: dnsmasq Service Fails to Start or Bind

Symptoms:

- Error on service start: "failed to bind interface"

Causes & Fixes:

- Port 53 already in use (conflict with `systemd-resolved` or `BIND`)
  Fix: Set the following in `/etc/dnsmasq.conf`:

  ```
  port=0
  ```

- Wrong interface or misconfigured systemd service
  Fix: Ensure `dnsmasq` is bound only to the internal interface (`enp2s0`), not Wi-Fi

---

Issue 3: DNS Resolution Fails for OpenShift Domains

Symptoms:

- Browser error: "We're having trouble finding that site"
- `nslookup` fails for OpenShift-related subdomains

Causes & Fixes:

- BIND zone file has syntax or A record issues
  Fix: Double-check `/var/named/ssm.dsw.ost.zone`
- DNS queries are not reaching BIND
  Fix: Ensure the following is present in `/etc/named.conf`:

```
listen-on port 53 { 127.0.0.1; 192.168.10.1; };
allow-query { any; };
```

- Client is not using `192.168.10.1` as DNS
  Fix: Ensure the following line exists in `dnsmasq.conf`:

```
dhcp-option=6,192.168.10.1
```

Issue 4: Clients Have No Internet Access

Symptoms:

- Clients get IP and DNS but can't ping external sites

Causes & Fixes:

- NAT not enabled between Ethernet and Wi-Fi
  Fix:

```
sudo iptables -t nat -A POSTROUTING -o wlp3s0 -j MASQUERADE
sudo iptables -A FORWARD -i enp2s0 -j ACCEPT
sudo service iptables save
```

- IP forwarding not enabled
  Fix:

```
echo "net.ipv4.ip_forward = 1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

Issue 5: DNS Zone Reload Fails or Invalid

Symptoms:

- `rndc reload` or `named-checkzone` throws error

Causes & Fixes:

- Serial number or syntax issue in zone file
  Fix: Ensure the serial number is valid and incrementing:

```
2025060401 ; Serial
```

- Missing NS record or bad TTL values
  Fix: Ensure required NS and A records are present with:

```
$TTL 86400
```

Issue 6: Service Doesn’t Start on Boot

Symptoms:

- DHCP or DNS not running after reboot

Fix:

```
sudo systemctl enable dnsmasq
sudo systemctl enable named
```

Issue 7: Residual DHCP Leases Cause Conflicts After Reconfiguration

Symptoms:

- Reused or incorrect IPs assigned
- `dnsmasq` seems to ignore config changes

Causes & Fixes:

- Old lease files persist in `/var/lib/NetworkManager/`
  Fix: Clear DHCP leases manually:

```
sudo systemctl stop NetworkManager
sudo rm /var/lib/NetworkManager/*.lease
# Or for specific interface:
sudo rm /var/lib/NetworkManager/*enp2s0.lease
sudo systemctl start NetworkManager
sudo reboot
```

Verify Cleanup:

```
ls /var/lib/NetworkManager/*.lease
```

Should return nothing (or only newly regenerated leases).

Summary

By combining `dnsmasq` for lightweight DHCP and `BIND` for advanced DNS, this setup enables a Fedora/RHEL system to serve as a capable gateway for OpenShift deployment. Static mapping, authoritative resolution, and external forwarders are all integrated.

Always verify from the client and debug using logs:

```
sudo journalctl -u dnsmasq
sudo journalctl -u named
```

Stay consistent with naming, interface bindings, and firewall rules for reliable operation.

Part 2: Creating the OpenShift Master Node

Installing OpenShift Container Platform with the Agent-based Installer

These steps guide you through deploying a Single Node OpenShift (SNO) using the Agent-based Installer in a disconnected environment.

Downloading the Agent-based Installer

1. Log in to the OpenShift Container Platform web console.
2. Go to: `Datacenter > Run Agent-based Installer locally`.
3. Select your OS and architecture.
4. Click **Download Installer** to download and extract it.
5. Click **Download pull secret** or **Copy pull secret** to get the pull secret.
6. Click **Download command-line tools** and ensure `openshift-install` is in your system `$PATH`.

⚙ Creating the Preferred Configuration Inputs

> ⚠ Use `install-config.yaml` and `agent-config.yaml` instead of ZTP manifests.

Step 1: Get the Correct Disk ID for installationDisk

```
ls -l /dev/disk/by-id/ | grep nvme0n1
```

Example Output:

```
lrwxrwxrwx 1 root root 13 Jun 5 16:30 wwn-0x50026b768372d4f8 -> ../../nvme0n1
```

Use the path: `/dev/disk/by-id/wwn-0x50026b768372d4f8` in your install-config.yaml.

> ⚠ Do not use the router or admin disk.

Step 2: Create a Working Directory

```
mkdir openshift
cd openshift
```

Step 3: Create `install-config.yaml`

```
apiVersion: v1
baseDomain: dsw.ost
compute:
- name: worker
  replicas: 0
controlPlane:
  name: master
  replicas: 1
metadata:
  name: ssm
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.10.0/16
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
bootstrapInPlace:
  installationDisk: /dev/disk/by-id/wwn-0x50026b768372d4f8
pullSecret: '<your_pull_secret>'
sshKey: |
  ssh-rsa AAAAB3...your-ssh-key-here...
```

Step 4: Create `agent-config.yaml`

```
apiVersion: v1alpha1
kind: AgentConfig
metadata:
  name: ssm
rendezvousIP: 192.168.10.11
```

Step 5: Generate the Bootable ISO

```
openshift-install --dir . agent create image
```

Step 6: Post-Boot Password Setup

After booting master node from ISO:

```
ssh core@<master-node-ip>
sudo passwd core
```

Part 3: Adding Worker Nodes Manually

Manually add worker nodes to an existing Single Node OpenShift (SNO) cluster using RHCOS ISO and the `worker.ign` file.

Prerequisites

- Single Node OpenShift installed
- `oc` CLI configured
- Cluster-admin access
- DNS (A and PTR) for new worker nodes

Step 1: Set OCP Version

```
OCP_VERSION=<ocp_version>  # e.g. latest-4.18
```

Step 2: Set Architecture

```
ARCH=x86_64  # or aarch64
```

Step 3: Retrieve `worker.ign`

```
oc extract -n openshift-machine-api secret/worker-user-data-managed --keys=userData --to=- > worker.ign
```

Step 4: Host the `worker.ign` File

```
python3 -m http.server 8080
```

Step 5: Download and Extract OpenShift Installer

```
curl -k https://mirror.openshift.com/pub/openshift-v4/clients/ocp/$OCP_VERSION/openshift-install-linux.tar.gz -o openshift-install-l
tar zxvf openshift-install-linux.tar.gz
chmod +x openshift-install
```

Step 6: Get the RHCOS ISO URL

```
ISO_URL=$(./openshift-install coreos print-stream-json | grep location | grep $ARCH | grep iso | cut -d\" -f4)
```

Step 7: Download the RHCOS ISO

```
curl -L $ISO_URL -o rhcos-live.iso
```

Step 8: Install RHCOS on Worker Node

After booting worker with `rhcos-live.iso`:

```
coreos-installer install --ignition-url=http://<host_ip>:8080/worker.ign /dev/nvme0n1
```

Then reboot:

```
reboot
```

Step 9: Approve CSRs

Manual Approval:

```
oc get csr
oc adm certificate approve <csr_name>
```

Auto-approval (⚠ for dev/test only):

```
watch -n 10 'oc get csr --no-headers | awk '\''$2 == "Pending" {print $1}'\'' | xargs -r oc adm certificate approve'
```

Step 10: Verify Worker Node Status

```
oc get nodes
```

Output should show new worker nodes as `Ready` and role as `worker`.
