You can treat the two Rocky 9 VMs as `rocky-91` (192.168.4.91) and `rocky-92`(192.168.4.192). Below are lab-style exercises; each block is one exercise.

Current setup: 192.168.4.198/24,192.168.4.1 - device enp1s0

Second interface enp9s0 192.168.5.198/24,192.168.4.1 – device
enp9s0

1. nmcli: basic static config between two hosts

Configure `rocky-91` and `rocky-92` with static addresses and verify connectivity.

Change the values to your actual values.

1. On `rocky-91`, list connections and identify the primary Ethernet connection name.
    - `nmcli con show`
    - `nmcli dev status`
2. Modify that connection so `rocky-91` uses:
    - IP: 192.168.5.191/24
    - Gateway: 192.168.4.1
    - DNS: 8.8.8.8
    - Connection autostart on boot.
    - Use `nmcli con modify enp9s0 ipv4.addresses` 192.168.5.191/24 `ipv4.gateway 192.168.4.1 ipv4.dns` 8.8.8.8 `ipv4.method manual connection.autoconnect yes`
3. Do the same on `rocky-92` with IP 192.168.5.192/24.
4. Reload and reactivate connections on both hosts using

    `nmcli con down` enp9s0

    `nmcli con up enp9s0`
5. From `rocky-91`, ping `192.168.5.192`; then from `rocky-92`, ping `192.168.5.191`.

2. nmcli: default route vs nondefault route

Create a dedicated route to another subnet in addition to the default route.

1. Assume a “remote lab” network 10.10.10.0/24 reachable via a router at 192.168.4.254.
2. On `rocky-91`, show current IPv4 routes: `ip route` and `nmcli dev show <iface>`.
3. Add a static route **only for** 10.10.10.0/24 via 192.168.4.254:
   - `nmcli con modify enp9s0 +ipv4.routes "10.10.10.0/24 192.168.4.254"`
   - Reload connection.
4. Verify that:
   - The default route is still via 192.168.4.1.
   - A more specific route for 10.10.10.0/24 via 192.168.4.254 now exists.
5. On `rocky-92`, do **not** configure this static route.
6. Use `ip route get 10.10.10.5` on both hosts and compare which gateway is chosen (on `rocky-92`, the kernel should fall back to the default route).

3. resolv.conf and nsswitch.conf: different name resolution paths

Explore how DNS and `/etc/hosts` interact.

1. On both hosts, inspect `/etc/resolv.conf` and identify the configured nameserver(s).
2. On `rocky-91`, edit `/etc/hosts`:
   - Add: `192.168.4.192 rocky-92.local rocky-92`
3. From `rocky-91`, test name resolution:
   - `ping -c1 rocky-92`
   - `getent hosts rocky-92`
4. On `rocky-91`, back up and edit `/etc/nsswitch.conf`:
   - Change the `hosts:` line to `hosts: files dns` (if not already).
   - Test `getent hosts rocky-92` again to confirm it still prefers `/etc/hosts`.

     (how is the response time?)
5. Now change that line on `rocky-91` to `hosts: dns files`, and run the same test.

   (how is the response
   time?)

4. hosts file only: "DNS outage" drill

Simulate DNS outage and fall back to static entries.

1. On `rocky-92`, note the current resolver (`/etc/resolv.conf`).
2. Intentionally misconfigure DNS:
   - Set nameserver to 192.0.2.1 (TEST-NET-1, not responding) or remove all nameserver lines.
3. Try `ping uadmin.org` and note the failure.
   - Add a nameserver back to /etc/resolv.conf

     ```
     echo nameserver 8.8.8.8 > /etc/resolv.conf
     ```
4. Check:

- `getent hosts uadmin.org` (should succeed).

5. firewalld + httpd: expose web from one host to the other

Serve a simple web page on `rocky-91` and consume it from `rocky-92`.

1. On `rocky-91`, install and enable Apache:
   - `dnf install httpd`
   - `systemctl enable --now httpd`
2. Verify local access:
   - `curl http://127.0.0.1/` (on `rocky-91`).
3. Check firewalld status and active zones:
   - `systemctl status firewalld`
   - `firewall-cmd --get-active-zones`
4. Run the curl command from rocky-92
5. On rocky-91 add the http service to the public zone.:
   - `firewall-cmd --zone=public --add-service=http`
   - `firewall-cmd --zone=public --add-service=http --permanent`
   - `firewall-cmd --reload`
6. From `rocky-92`, access:
   - `curl http://192.168.4.191/`
7. Tighten firewall on `rocky-91`:
   - Remove HTTP from the zone (runtime), test from `rocky-92` again and observe the timeout.
   - Re-add HTTP and confirm it works.

6. firewalld: host-only access restriction

Allow HTTP on `rocky-91` only from `rocky-92`, block everyone else.

1. On `rocky-91`, ensure HTTP is installed and working as in exercise 5.
2. Identify the relevant zone (likely `public`) and interface: `firewall-cmd --get-active-zones`.
3. Remove generic HTTP allowance:
   - `firewall-cmd --zone=public --remove-service=http --permanent`
   - `firewall-cmd --reload`
4. Add a rich rule allowing HTTP only from 192.168.4.192:
   - `firewall-cmd --zone=public --add-rich-rule='rule family="ipv4" source address="192.168.4.192" service name="http" accept' --permanent`
   - `firewall-cmd --reload`
5. From `rocky-92`, access `http://192.168.4.91/` (should work).
6. From a **third** host on the same subnet (if available), attempt to access and verify it fails.

7. Interface naming and predictability

Explore current interface naming and configure a stable custom name.

1. On both hosts, list all interfaces:
   - `ip link`
   - Note names such as `ens160`, `enp1s0`, etc.
2. On `rocky-91`, inspect udev and systemd-udevd rules for interfaces:
   - `udevadm info /sys/class/net/<iface>`
3. Create a systemd `.link` file in `/etc/systemd/network/` on `rocky-91` so that the primary NIC becomes `lan0`:
   - Match on the current name or MAC address.
   - Set `Name=lan0`.
4. Reboot `rocky-91` and confirm that:

- The interface is now called `lan0`.
- The existing nmcli connection has either been updated or you create a new one bound to `lan0`.

5. Adjust your `nmcli` connection to use `lan0` and confirm that connectivity between `rocky-91` and `rocky-92` still works.