**Scenario:** You want to add an additional static IPv4 and IPv6 address to your RHEL 9 EC2 instance using nmcli.

**Steps:**

1. **Launch an RHEL 9 EC2 Instance (Primary ENI via DHCP):**
   * Go to the AWS EC2 Dashboard and click "Launch instance".
   * **Name and tags:** Give it a meaningful name (e.g., rhel-network-test).
   * **Application and OS Images (Amazon Machine Image - AMI):** Select "Red Hat Enterprise Linux" and choose an RHEL 9 AMI (e.g., "Red Hat Enterprise Linux 9 (HVM) - Kernel 5.14").
   * **Instance type:** A t2.micro or t3.micro is usually sufficient for testing.
   * **Key pair (login):** Select an existing key pair or create a new one to SSH into the instance.
   * **Network settings:**
     + **VPC:** Choose an existing VPC or create a new one.
     + **Subnet:** Select a subnet that has **both IPv4 and IPv6 CIDR blocks** associated with it. This is crucial for IPv6 testing. If your subnet doesn't have IPv6, you'll need to add an IPv6 CIDR block to your VPC and then associate it with your subnet.
     + **Auto-assign public IP:** Enable it if you want direct internet access via public IP.
     + **Firewall (security groups):** Ensure you have a security group that allows **SSH (port 22)** from your IP address. For testing later, you might want to add ICMP (ping) rules for IPv4 and IPv6.
   * **Review and Launch:** Launch the instance.
2. **Create a New Elastic Network Interface (ENI) in the AWS Console:** This ENI will host your static IPv4 and IPv6 addresses.
   * Go to the EC2 Dashboard -> **Network & Security** -> **Network Interfaces**.
   * Click **Create network interface**.
   * **Description:** Give it a meaningful name (e.g., rhel-secondary-eni).
   * **Subnet:** **Crucially, select the SAME subnet where your EC2 instance is running.**
   * **Security Groups:** Attach the **same security group** (or one with equivalent rules) as your EC2 instance. This is important so traffic on the secondary interface is allowed.
   * **IPv4 Private IP:** Under "IPv4 private IP addresses", click "Auto-assign" or "Custom" to pick an available IP from the subnet's IPv4 range. Let's assume AWS assigns 172.31.32.100.
   * **IPv6 IP:** Click "Add new IPv6 address". AWS will typically assign one automatically from the subnet's IPv6 CIDR block (e.g., 2600:1f18:222a:3f01::/64). Note this address down. Example: 2600:1f18:222a:3f01:b3d2:ac78:c9f0:1234.
   * Click **Create network interface**.
3. **Attach the New ENI to Your RHEL EC2 Instance:**
   * In the Network Interfaces list, select your newly created ENI.
   * Go to **Actions** -> **Attach**.
   * Select your running RHEL EC2 instance from the dropdown.
   * **Device index:** Choose 1 (for eth1). This will usually make it eth1 or enp0sX (where X is incremented) in the OS.
   * Click **Attach network interface**.
4. **Configure the New ENI within the RHEL 9 Instance (using nmcli):**
   * **SSH into your RHEL 9 EC2 instance** using its primary public IP.
   * **Verify the new interface is detected:**

Bash

nmcli device status

# Look for a new device that is 'unmanaged'. It might be named eth1, enp0s4, etc.

# Example output:

# DEVICE TYPE STATE CONNECTION

# enp0s3 ethernet connected System enp0s3

# eth1 ethernet unmanaged --

From this point, let's assume the secondary interface name is eth1.

* + **Create a new NetworkManager connection profile for eth1 with static IPv4 and IPv6:**
    - **Important:** Replace 172.31.32.100/20 and 2600:1f18:222a:3f01:b3d2:ac78:c9f0:1234/64 with the *actual IP addresses* that AWS assigned to your secondary ENI in step 2.
    - **Gateways (ipv4.gateway and ipv6.gateway) should be left empty (or set to "") for secondary ENIs.** The default route for your instance goes through the primary ENI (enp0s3 in this example).
    - **DNS:** Use your VPC's DNS resolver for IPv4 (e.g., 172.31.0.2 for a 172.31.0.0/16 VPC). For IPv6, you can use public DNS like Google's.

Bash

sudo nmcli connection add type ethernet con-name static-eth1 ifname eth1 \

ipv4.method manual \

ipv4.addresses 172.31.32.100/20 \

ipv4.gateway "" \

ipv4.dns "172.31.0.2" \

ipv6.method manual \

ipv6.addresses 2600:1f18:222a:3f01:b3d2:ac78:c9f0:1234/64 \

ipv6.gateway "" \

ipv6.dns "2001:4860:4860::8888 2001:4860:4860::8844" \

connection.autoconnect yes

* + **Activate the new connection:**

Bash

sudo nmcli connection up static-eth1

This command will bring up the static-eth1 connection profile on the eth1 interface.

1. **Verify Configuration:**
   * **Check IP addresses:**

Bash

ip a

# Look for 'eth1' and confirm it has both your static IPv4 and IPv6 addresses.

* + **Check active connections:**

Bash

nmcli connection show --active

# Confirm 'static-eth1' is listed as active.

* + **Test connectivity (IPv4 and IPv6):**

Bash

ping -I eth1 google.com # Tests IPv4 using the new interface

ping -6 -I eth1 google.com # Tests IPv6 using the new interface

# You might need to install `iputils` for ping, or `bind-utils` for dig.

# sudo dnf install iputils bind-utils -y

dig google.com @172.31.0.2 # Test IPv4 DNS

dig -6 google.com @2001:4860:4860::8888 # Test IPv6 DNS

* + **Check routing table (optional, but good for understanding):**

Bash

ip r # Check IPv4 routes (default should still be via primary ENI)

ip -6 r # Check IPv6 routes (default should still be via primary ENI)

You'll typically only see local routes for eth1 because its purpose is to provide additional IPs, not a separate default route.

**Why this approach works and is safe:**

* **Primary ENI Intact:** Your main enp0s3 (or eth0) interface remains on DHCP, allowing the instance to communicate with AWS services and maintaining your SSH connection.
* **AWS-Allocated IPs:** You are configuring static IPs that AWS has *actually allocated* to that specific secondary ENI, avoiding conflicts.
* **Correct Routing Model:** Secondary ENIs are for additional IP addresses, not for establishing a separate default route. The primary ENI handles the instance's default route

Create a network manager connection on "machine" name "eth0" that manages ens3 interface

Configure the following static ip

ipv4 172.138.1.1/20

this should be from m/c range

ipv6 ddd

--------------------------

ip a

nmcli connection show

find name ens3 (or equivalent in your m/c)

nmcli connection add type ethernet con-name eth0 ifname ens3

ip a

find the ip range you should try

nmcli connection modify eth0 ipv4.address 172.31.32.7/20 ipv4.method manual connection.autoconnect yes

similar comman can modify ipv6 also

nmcli connection show

nmcli connection up eth0

nmcli connection show

ip a

check the ip address configured