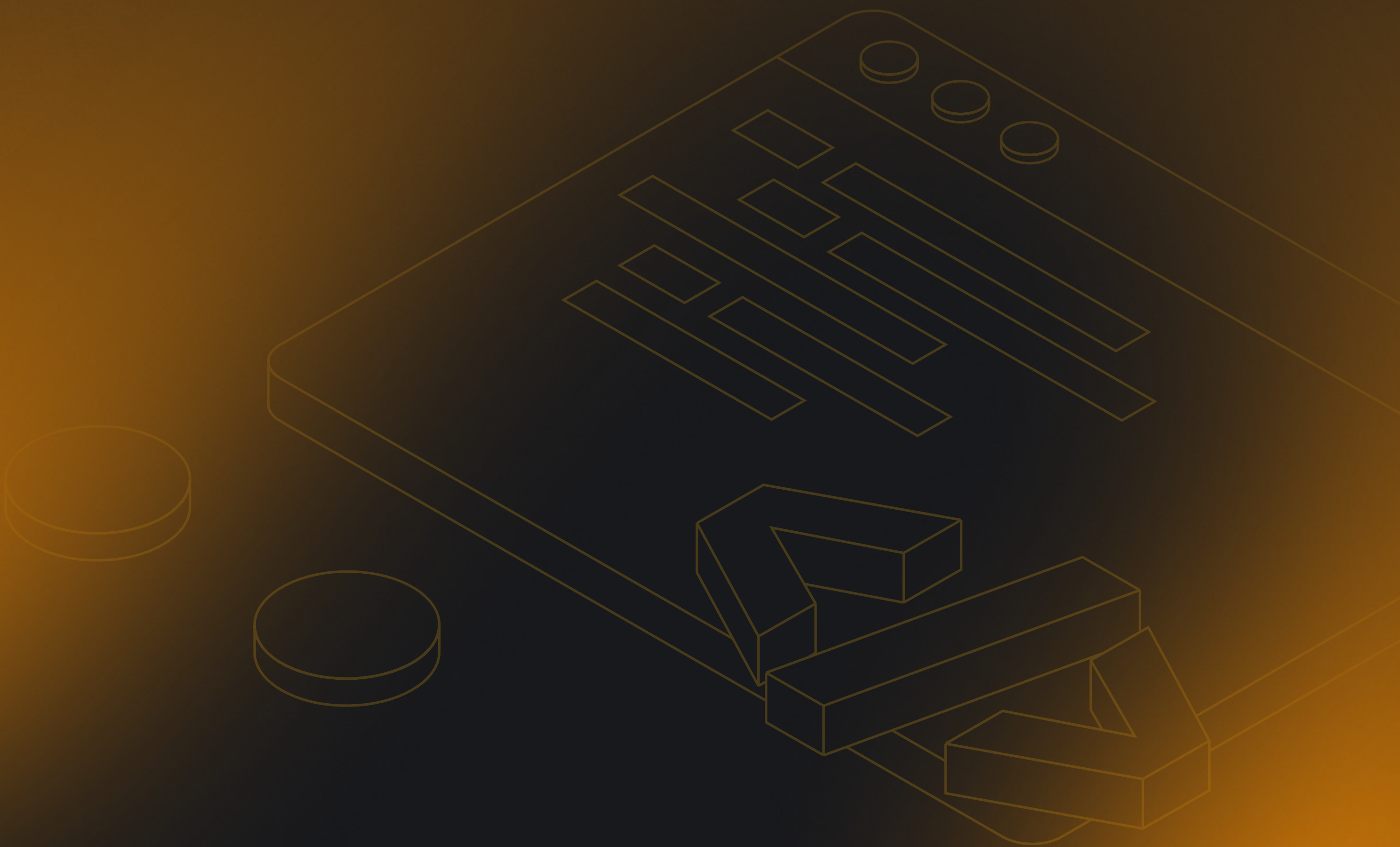
VyOS Networks



Deployment Guide | Technical Doc

DEPLOYING VYOS IN AWS AS A NAT INSTANCE

November 2025

Index:

Deploying VyOS in AWS as a NAT Instance 3

VyOS as a NAT Instance 3

Lab Overview 3

Topology 3

Network Design 4

Test Objectives 4

Configuration 5

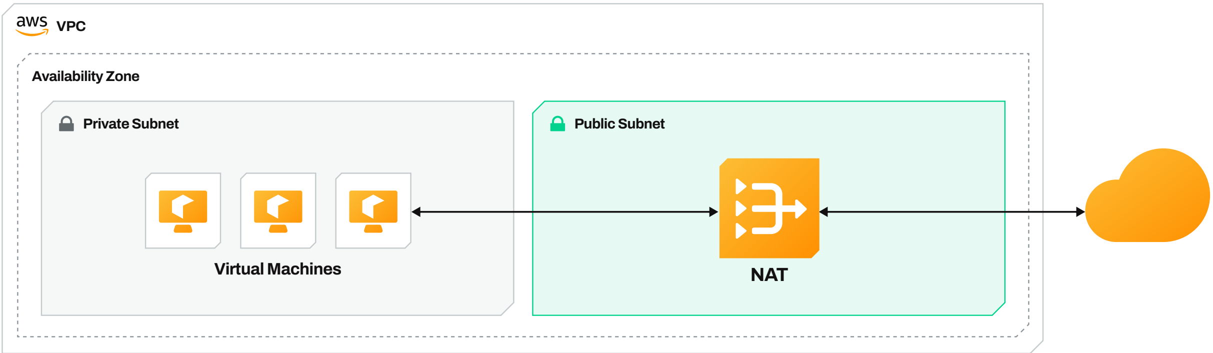
1. Launch the VyOS instance 5
2. Add the second (LAN) interface 6
3. Connect to VyOS instance 7
4. Configure VyOS for NAT 8
5. Configure VyOS firewall 8
6. Configure AWS subnet routing 10
7. Test connectivity 11

Appendix 14

Steps to Create a Public Subnet in AWS 14

Deploying VyOS in AWS as a NAT Instance

When designing network architectures in AWS, securely enabling internet access for resources in private subnets is a common requirement. To achieve this, AWS provides two primary solutions: the NAT Gateway and the NAT Instance. Both allow outbound internet traffic from private instances while blocking unsolicited inbound connections, but they differ significantly in terms of cost, scalability, management, and flexibility.



VyOS as a NAT Instance

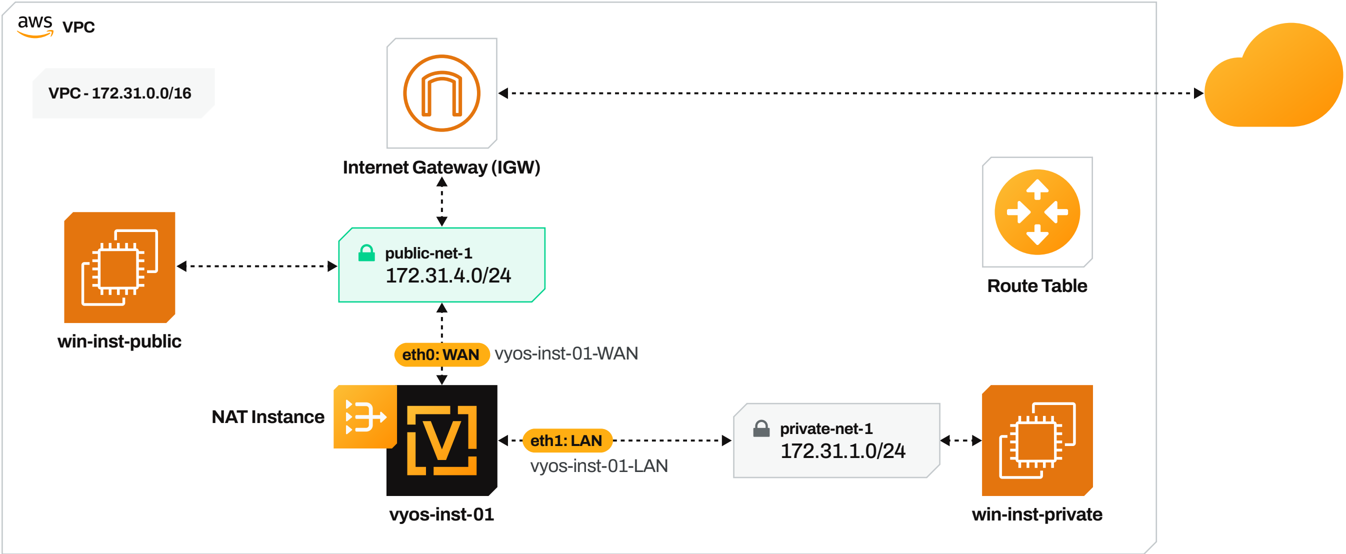
An AWS NAT Instance is a virtual machine configured to provide Network Address Translation (NAT) for instances in a private subnet. It allows these instances to initiate outbound traffic to the internet or other AWS services, while preventing unsolicited inbound connections. Deploying VyOS on AWS (via EC2) enables enterprises to bring advanced, flexible network services into their cloud infrastructure.

Lab Overview

To test this functionality, we will deploy a lab environment consisting of three instances on AWS: two Windows EC2 instances and one VyOS router instance.
The setup will emulate a typical enterprise network with private and public subnets, routing, and NAT.

Topology

- **VyOS Instance** – Acts as the router/firewall between the LAN and the Internet. It provides connectivity, NAT, routing and firewalling between subnets.
- **Windows Instance 1 (win-inst-private)** – Located in the private subnet, representing an internal user or server without direct Internet access.
- **Windows Instance 2 (win-inst-public)** – Located in the public subnet, acting as a jump server that provides access to the internal server, which has no direct Internet connectivity.



Network Design

- **Public Subnet:** Connected to the Internet Gateway. The VyOS WAN interface and the public Windows instance reside here.
- **Private Subnet:** Connected to the VyOS LAN interface. The private Windows instance is here.
- **Routing:**
  - The private subnet's default route (0.0.0.0/0) points to the VyOS LAN interface.
  - VyOS performs SNAT for outbound traffic and allows controlled inbound traffic (e.g., SSH, RDP, ICMP) and outbound traffic if necessary.

Test Objectives

- **Verify routing and connectivity between LAN and WAN through VyOS.**
- **Validate SNAT functionality for Internet access from the private subnet.**
- **Test internet access from the private subnet.**

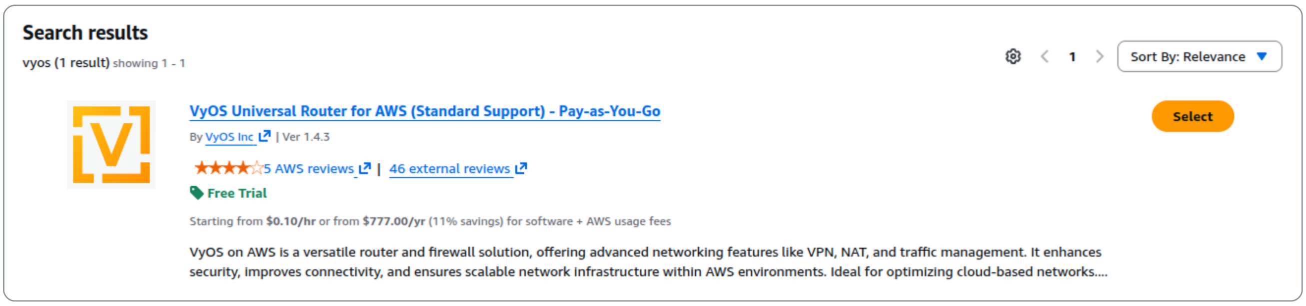
Configuration

1. Launch the VyOS instance

1. Go to the AWS Management Console > EC2 > Launch Instance.

2. Choose a VyOS AMI:

Search for "VyOS" and select AWS Marketplace AMIs.

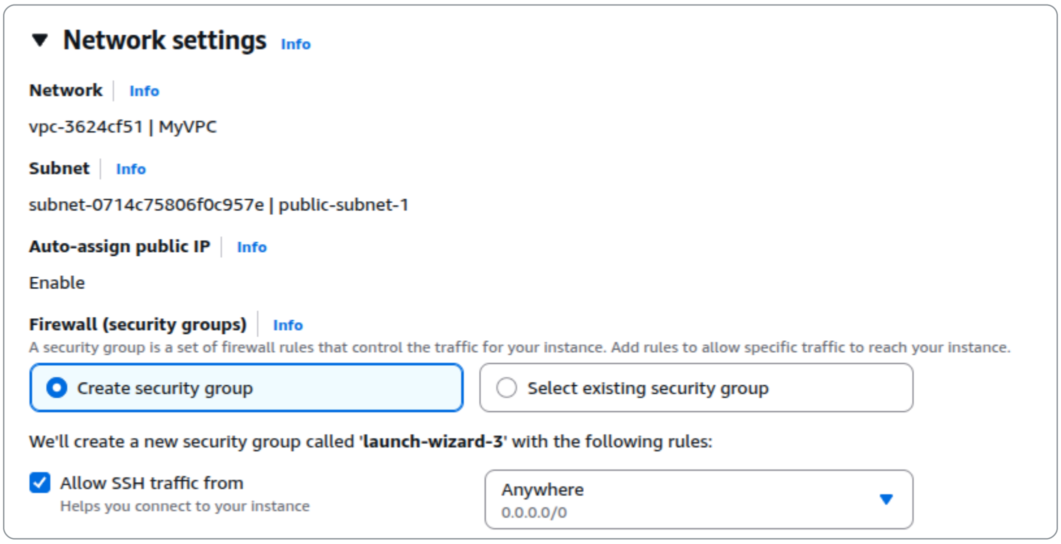


3. Select an Instance Type (e.g., `t3.medium` is a good starting point).

> ⚠️ Create a new key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

4. In the Network Settings section:

- Choose your **VPC.**
- Select a **public subnet** for the **WAN interface** (this one should have Internet Gateway access).
- **Enable Auto-assign Public IP.**
- Create or select a security group to allow SSH access.



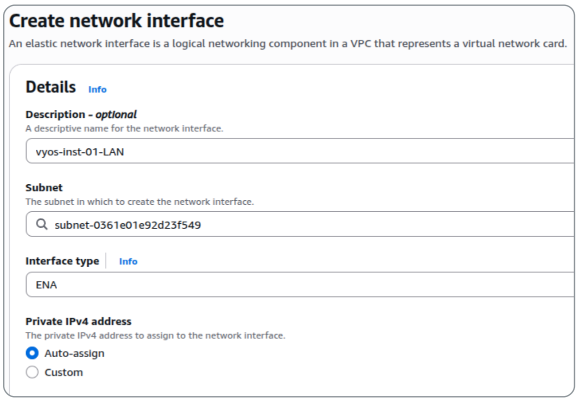
2. Add the second (LAN) interface

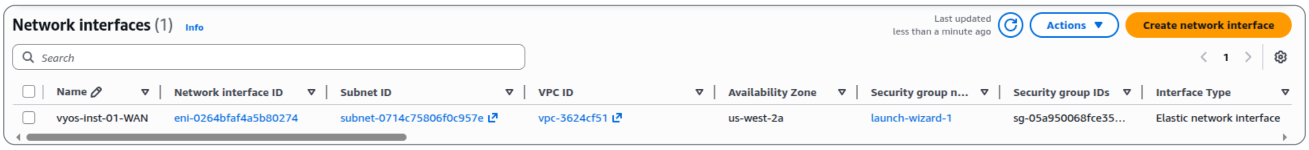
1. Once the instance is launched, go to EC2 > Network Interfaces > Create network interface.

2. Choose:

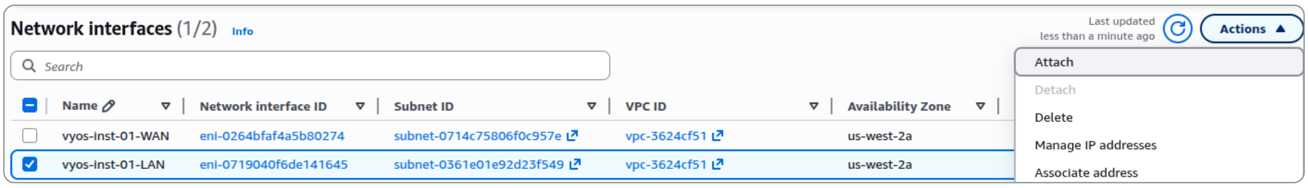
- The **same VPC.**
- A **private subnet** for the LAN side (without Internet access).

3. Assign a private IP (optional).



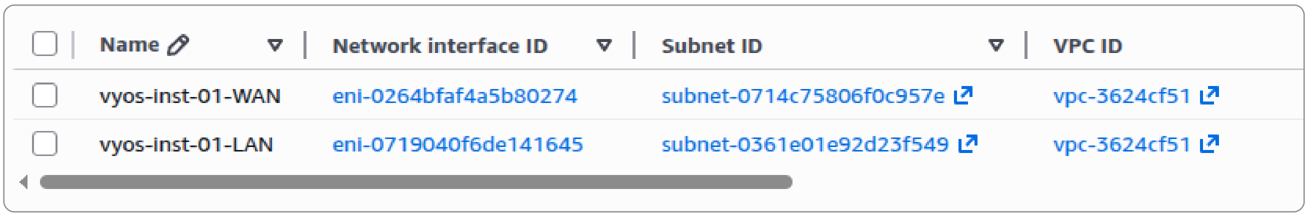


4. Select the ENI > Actions > Attach to instance > choose the VyOS instance.

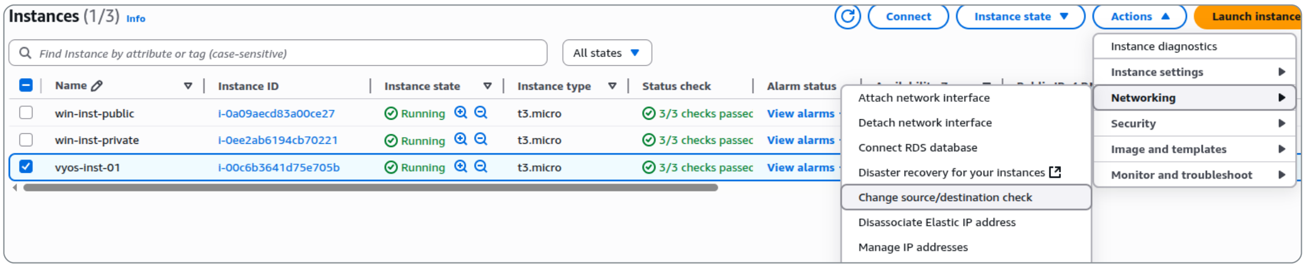


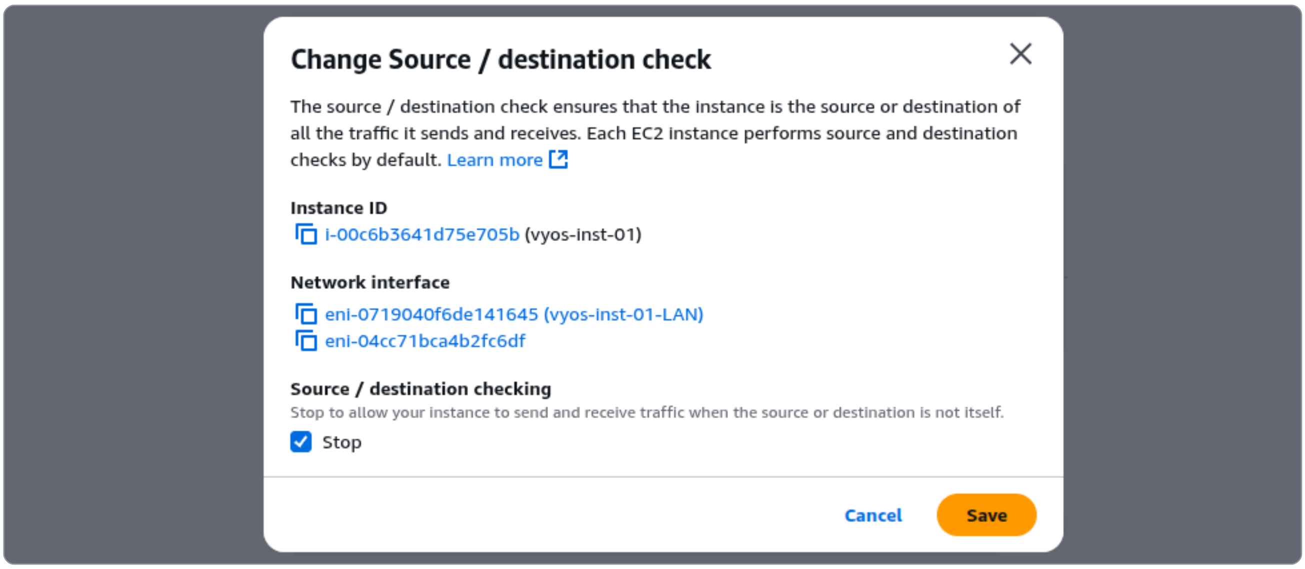
Now your instance has:

- **eth0 >** WAN (public subnet)
- **eth1 >** LAN (private subnet)



5. **Finally, go to Instances > select VyOS instance > Actions > Networking > Change source/ destination check, select Stop**

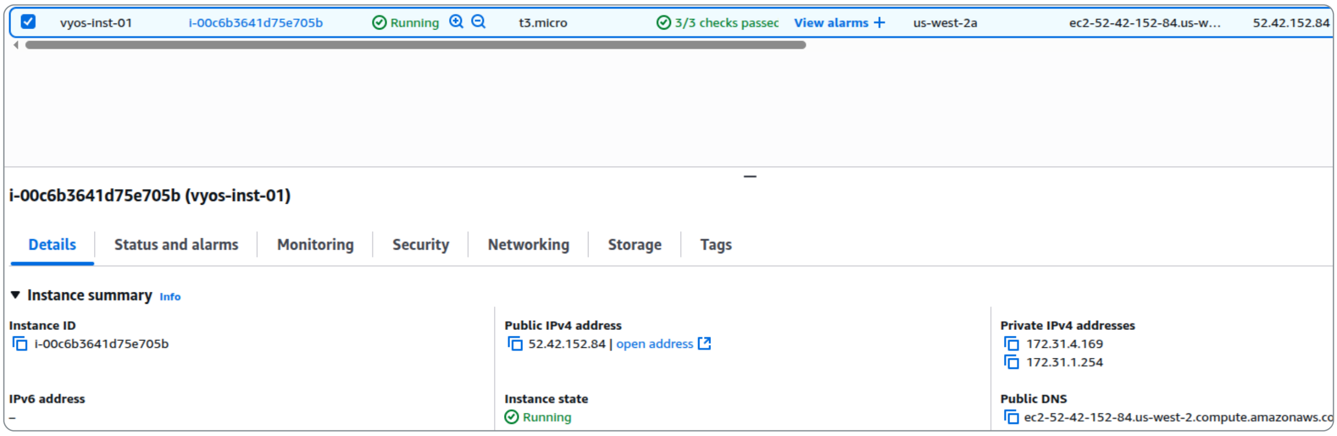




The **Source/Destination Check** determines if an EC2 instance can forward traffic not intended for itself, it must be disabled for routing or NAT functions.

3. Connect to VyOS instance

SSH into the VyOS instance using its public IP:



1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is "vyos-key.pem" for our example.
3. Run this command, if necessary, to ensure your key is not publicly viewable: chmod 400 "vyos-key.pem"
4. Connect to your instance using its Public DNS: ec2-44-249-142-84.us-west-2.compute.amazonaws.com (for our example)

```
ssh -i "vyos-key.pem" vyos@ec2-35-85-54-70.us-west-2.compute.amazonaws.com
```

```
Welcome to VyOS!

   . VyOS 1.4.2
              sagitta

 * Documentation:  https://docs.vyos.io/en/sagitta
 * Project news:   https://blog.vyos.io
 * Bug reports:    https://vyos.dev

You can change this banner using "set system login banner post-login" command.

VyOS is a free software distribution that includes multiple components,
you can check individual component licenses under /usr/share/doc/*/copyright
vyos@ip-172-31-4-226:~$
```

4. Configure VyOS for NAT

```
# Assign IP addresses and system hostname
set interfaces ethernet eth0 description 'WAN'
set interfaces ethernet eth1 description 'LAN'
set interfaces ethernet eth1 address dhcp
set system host-name VyOS-NAT

# Enable NAT for LAN -> WAN
set nat source rule 100 outbound-interface name 'eth0'
set nat source rule 100 source address '172.31.1.0/24'
set nat source rule 100 translation address 'masquerade'

# Commit and save
commit
save
```

5. Configure VyOS firewall

In this example, we will create two interface groups — a **WAN** group for our interfaces connected to the public internet and a **LAN** group for the interfaces connected to our internal network. Additionally, we will create a network group, **NET-INSIDE-v4**, that contains our internal subnet.

We configure stateful connection filtering by creating rules on the base forwarding and input hook. We can block all other incoming traffic addressed to our local network.

We create a new chain **(OUTSIDE-IN)** which will drop all traffic that is not explicitly allowed at some point in the chain. Then, we can jump to that chain from the **forward** hook when traffic is coming from the **WAN** interface group and is addressed to our local network.

We configure access to the router itself, allowing SSH access from the inside/LAN network and rate limiting SSH access from the outside/WAN network.

First, create a new dedicated chain **(VyOS_MANAGEMENT)** for management access, which returns to the parent chain if no action is taken. Add a rule to accept traffic from the **LAN** interface group. Configure a rule on the **input** hook filter to jump to the **VyOS_MANAGEMENT** chain when new connections are addressed to port 22 (SSH) on the router itself. Finally, configure the **VyOS_MANAGEMENT** chain to accept connection from the **LAN** interface group while limiting requests coming from the **WAN** interface group to 4 per minute. Then we're allowing the router to respond to pings. Finally, we can now configure access to the services running on this router, allowing all connections coming from localhost.

```
# Create firewall groups
set firewall group interface-group LAN interface 'eth1'
set firewall group interface-group WAN interface 'eth0'
set firewall group network-group NET-INSIDE-v4 network '172.31.1.0/24'

# Create ipv4 forward rules
set firewall ipv4 forward filter rule 5 action 'accept'
set firewall ipv4 forward filter rule 5 state 'established'
set firewall ipv4 forward filter rule 5 state 'related'
set firewall ipv4 forward filter rule 10 action 'drop'
set firewall ipv4 forward filter rule 10 state 'invalid'
set firewall ipv4 forward filter rule 100 action 'jump'
set firewall ipv4 forward filter rule 100 destination group network-group 'NET-INSIDE-v4'
set firewall ipv4 forward filter rule 100 inbound-interface group 'WAN'
set firewall ipv4 forward filter rule 100 jump-target 'OUTSIDE-IN'

# Create ipv4 input rules
set firewall ipv4 input filter default-action 'drop'
set firewall ipv4 input filter rule 5 action 'accept'
set firewall ipv4 input filter rule 5 state 'established'
set firewall ipv4 input filter rule 5 state 'related'
set firewall ipv4 input filter rule 10 action 'drop'
set firewall ipv4 input filter rule 10 state 'invalid'
set firewall ipv4 input filter rule 20 action 'jump'
set firewall ipv4 input filter rule 20 destination port '22'
set firewall ipv4 input filter rule 20 jump-target 'VyOS_MANAGEMENT'
set firewall ipv4 input filter rule 20 protocol 'tcp'
set firewall ipv4 input filter rule 30 action 'accept'
set firewall ipv4 input filter rule 30 icmp type-name 'echo-request'
set firewall ipv4 input filter rule 30 protocol 'icmp'
set firewall ipv4 input filter rule 30 state 'new'
set firewall ipv4 input filter rule 50 action 'accept'
set firewall ipv4 input filter rule 50 source address '127.0.0.0/8'

# Create ipv4 input rules
set firewall ipv4 name OUTSIDE-IN default-action 'drop'
set firewall ipv4 name VyOS_MANAGEMENT default-action 'return'
set firewall ipv4 name VyOS_MANAGEMENT rule 15 action 'accept'
set firewall ipv4 name VyOS_MANAGEMENT rule 15 inbound-interface group 'LAN'
set firewall ipv4 name VyOS_MANAGEMENT rule 20 action 'drop'
set firewall ipv4 name VyOS_MANAGEMENT rule 20 inbound-interface group 'WAN'
```

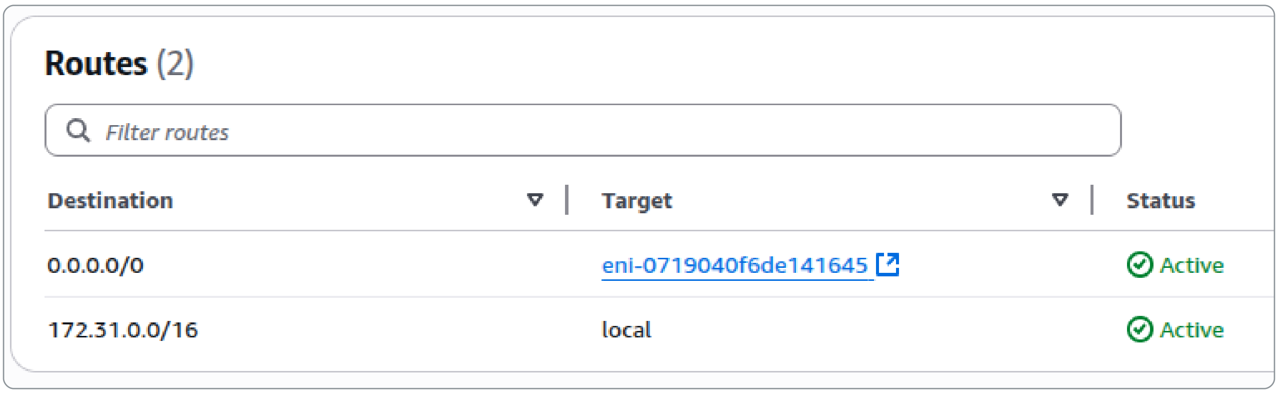
```
set firewall ipv4 name VyOS_MANAGEMENT rule 20 recent count '4'
set firewall ipv4 name VyOS_MANAGEMENT rule 20 recent time 'minute'
set firewall ipv4 name VyOS_MANAGEMENT rule 20 state 'new'
set firewall ipv4 name VyOS_MANAGEMENT rule 21 action 'accept'
set firewall ipv4 name VyOS_MANAGEMENT rule 21 inbound-interface group 'WAN'
set firewall ipv4 name VyOS_MANAGEMENT rule 21 state 'new'
```

6. Configure AWS subnet routing

In **AWS, route tables** define how network traffic is directed within a **VPC**. They tell each **subnet** where to send packets based on their destination IP address. Each subnet in a VPC must be associated with one route table.

Go to VPC dashboard:

- **Route Tables: Create route table for private subnet**, set the **default route (0.0.0.0/0)** to point to the **LAN interface's private IP** of the VyOS instance.

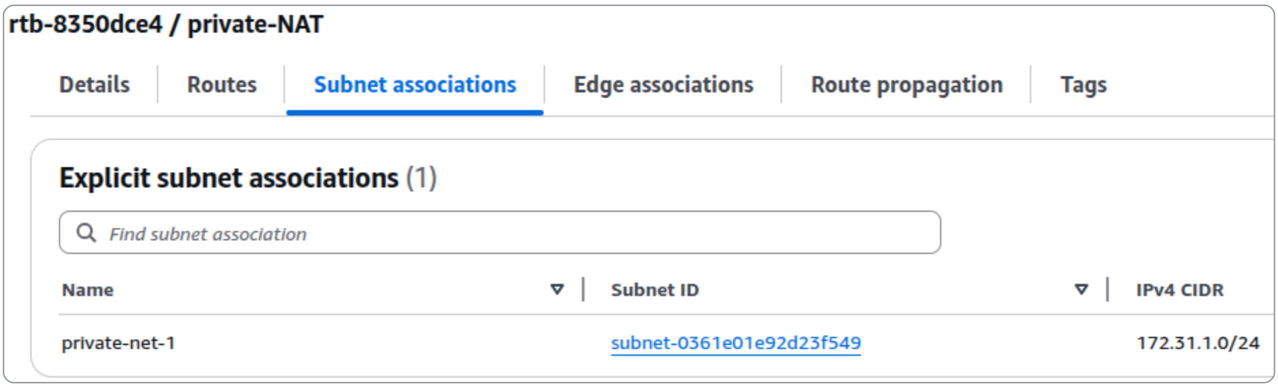


Routes (2)

Filter routes

Destination	Target	Status
0.0.0.0/0	eni-0719040f6de141645	Active
172.31.0.0/16	local	Active

- In **Subnet Associations** tab add the private subnet:



rtb-8350dce4 / private-NAT

Details | Routes | Subnet associations | Edge associations | Route propagation | Tags

Explicit subnet associations (1)

Find subnet association

Name	Subnet ID	IPv4 CIDR
private-net-1	subnet-0361e01e92d23f549	172.31.1.0/24

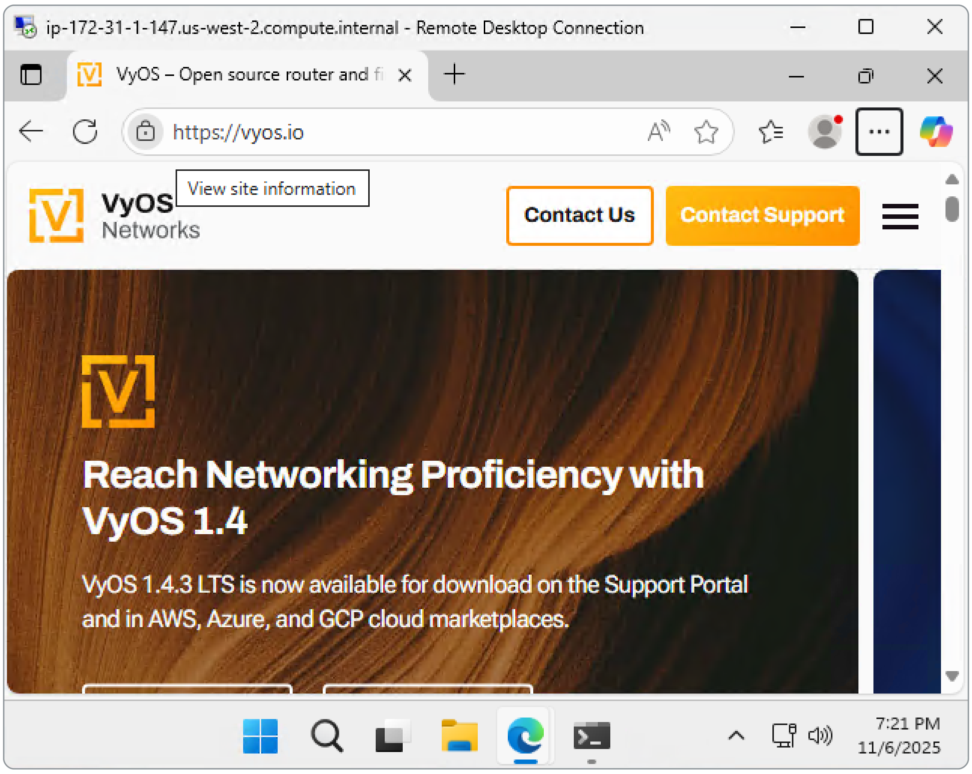
7. Test connectivity

- Launch another instance in the **private subnet** (private-net-1).
- Test Internet access (e.g., `ping 8.8.8.8`).

```
C:\Users\Administrator\ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

  Connection-specific DNS Suffix  . : us-west-2.compute.internal
  Link-local IPv6 Address . . . . . : fe80::93bd:e84b:8acd:5f63%6
  IPv4 Address. . . . . . . . . . . : 172.31.1.147
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . : 172.31.1.1
```

```
C:\Users\Administrator\ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=6ms TTL=116
Reply from 8.8.8.8: bytes=32 time=6ms TTL=116
Reply from 8.8.8.8: bytes=32 time=6ms TTL=116
Reply from 8.8.8.8: bytes=32 time=6ms TTL=116

Ping statistic for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 6ms, Average = 6ms
```

- Test internet access



- Show NAT translations at VyOS instance

```
vyos@VyOS-NAT# run show nat source translations

Pre-NAT             Post-NAT            Proto    Timeout    Mark    Zone
------------------  ------------------  -------  ---------  ------  ------
172.31.1.147:56992  172.31.4.169:56992  udp      27         0
172.31.1.147:50846  172.31.4.169:50846  tcp      38         0
172.31.1.147:61507  172.31.4.169:61507  tcp      115        0
172.31.1.147:54389  172.31.4.169:54389  tcp      40         0
172.31.1.147:53136  172.31.4.169:53136  tcp      20         0
172.31.1.147:63203  172.31.4.169:63203  tcp      115        0
172.31.1.147:62204  172.31.4.169:62204  tcp      38         0
172.31.1.147:50452  172.31.4.169:50452  tcp      115        0
172.31.1.147:61482  172.31.4.169:61482  tcp      45         0
172.31.1.147:49349  172.31.4.169:49349  udp      100        0
172.31.1.147:65360  172.31.4.169:65360  udp      2          0
172.31.1.147:54374  172.31.4.169:54374  tcp      95         0
172.31.1.147:55622  172.31.4.169:55622  udp      87         0
172.31.1.147:56663  172.31.4.169:56663  tcp      95         0
172.31.1.147:52596  172.31.4.169:52596  tcp      40         0
172.31.1.147:54053  172.31.4.169:54053  tcp      299        0
172.31.1.147:59469  172.31.4.169:59469  tcp      40         0
172.31.1.147:54631  172.31.4.169:54631  udp      2          0
172.31.1.147:61749  172.31.4.169:61749  tcp      115        0
172.31.1.147:51429  172.31.4.169:51429  tcp      65         0
172.31.1.147:50879  172.31.4.169:50879  tcp      95         0
172.31.1.147:62231  172.31.4.169:62231  tcp      2          0
```

```
vyos@VyOS-NAT# run show firewall statistics
Rulesets Statistics

---------------------------------
ipv4 Firewall "forward filter"

Rule       Packets    Bytes  Action    Source    Destination    Inbound-Interface    Outbound-interface
-------  ---------  -------  --------  --------  -------------  -------------------  --------------------
5            10120  9500114  accept    any       any            any                  any
10              28     1120  drop      any       any            any                  any
100              0        0  jump      any       NET-INSIDE-v4  WAN                  any
default        792   185117  accept    any       any            any                  any

---------------------------------
ipv4 Firewall "input filter"

Rule       Packets    Bytes  Action    Source       Destination    Inbound-Interface    Outbound-interface
-------  ---------  -------  --------  -----------  -------------  -------------------  --------------------
5              323    30313  accept    any          any            any                  any
10              14      588  drop      any          any            any                  any
20               1       60  jump      any          any            any                  any
30               1       60  accept    any          any            any                  any
50              98     7202  accept    127.0.0.0/8  any            any                  any
default         64     3574  drop      any          any            any                  any

---------------------------------
ipv4 Firewall "name OUTSIDE-IN"

Rule       Packets    Bytes  Action    Source    Destination    Inbound-Interface    Outbound-interface
-------  ---------  -------  --------  --------  -------------  -------------------  --------------------
default          0        0  drop      any       any            any                  any
```

```
---------------------------------
ipv4 Firewall "name VyOS_MANAGEMENT"

Rule       Packets    Bytes  Action    Source    Destination    Inbound-Interface    Outbound-interface
-------  ---------  -------  --------  --------  -------------  -------------------  --------------------
15               0        0  accept    any       any            LAN                  any
20               0        0  drop      any       any            WAN                  any
21               1       60  accept    any       any            WAN                  any
default          0        0  return    any       any            any                  any

[edit]
vyos@VyOS-NAT#
```

Appendix

Steps to Create a Public Subnet in AWS

1. **Open the VPC Console**

   Go to the AWS Management Console and open the **VPC** service.

2. **Select Your VPC**

   If you already have a VPC, select it. Otherwise, create a new one with an appropriate CIDR block.

3. **Create the Subnet**

   - Click **Subnets > Create subnet.**
   - Choose your **VPC**.
   - Enter a **name tag** (for example, `public-subnet-1`).
   - Select an **Availability Zone.**
   - Specify the **CIDR block** for the subnet.
   - Click **Create subnet**.

4. **Enable Auto-Assign Public IP**

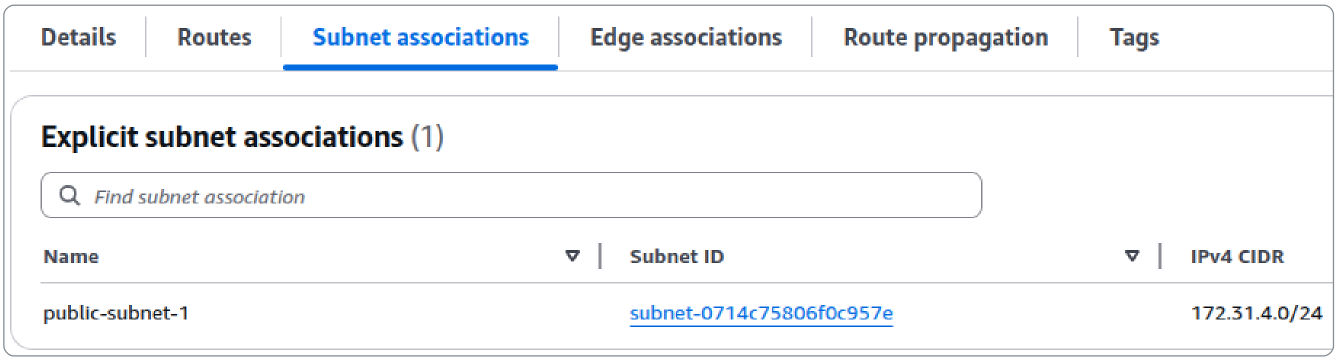
   - Select the subnet you just created.
   - Choose **Actions > Edit subnet settings**.
   - Enable **Auto-assign IP settings > Enable auto-assign public IPv4 address**.
   - Save the changes.

5. **Attach an Internet Gateway (IGW)**

   - Go to **Internet Gateways** and create a new one (or use an existing IGW).
   - Attach the IGW to your VPC.

6. **Create and Update the Route Table**

   - Go to **Route Tables** and either create a new one or select the default one associated with your VPC.
   - Edit the **Routes** and add a new route:
     - **Destination:** `0.0.0.0/0`
     - **Target:** your Internet Gateway (IGW).
   - Associate this route table with your **Public Subnet.**



After these steps, your subnet becomes a **Public Subnet**, because it can route traffic to and from the internet through the Internet Gateway.