How to deploy VyOS in Containerlab

What is Containerlab?

Containerlab is an orchestration and management tool for container-based networking labs. Using containers Containerlab builds topology based on the pre-defined information and creates virtual connections between specified nodes. In comparison with other lab environments, Containerlab provides fast deployment with high scalability, management and integration possibilities with other systems.

Official website: https://containerlab.srlinux.dev/

Environment used for this demonstration

Host parameters - Windows 10 OS with 16G RAM and 4 core CPU Corei5 9th Gen
Hypervisor - VMware Workstation Player 16.2.0
VM - Debian 10.11 fresh install with 8G RAM and 4 vCPU, Intel VT-x/EPT enabled
Docker version - 20.10.10 Stable
Containerlab version - 0.19.2 Stable
VyOS version - 1.3.0 Stable

> **Notice:** Same instruction has been successfully used to deploy labs using VyOS versions 1.2.x and 1.4.x

Adding VyOS Docker image to Containerlab

Pulling VyOS image from Docker Hub

If you have an active subscription with access to the VyOS images on Docker Hub, you can authenticate with personal credentials and pull pre-built LTS images following the next steps:

1. Authenticate with your docker hub credentials on the local machine where clab will run using the “docker login” command

2. Select an image from the available list and download it using the appropriate command (1.3.0 as an example below):

```
root@deb10:~# docker pull vyos/vyos:1.3.0
1.3.0: Pulling from vyos/vyos
0f7595eedb0e: Pull complete
Status: Downloaded newer image for vyos/vyos:1.3.0
docker.io/vyos/vyos:1.3.0
```

Notice: If you don't have an active subscription with access to the LTS images you can use development builds from https://hub.docker.com/r/vyos/image/tags

Building Docker container from ISO example

Notice: We recommend using the latest stable version of VyOS to get the best experience from the deployment, but it's up to you to choose the specific version for tests in the lab.

1. Download VyOS desired release ISO image to the host device running Containerlab

```
root@deb10# wget https:<url>/vyos-<version>-amd64.iso
```

2. Create directory and mount ISO image

```
root@deb10# mkdir rootfs
root@deb10# mount -t iso9660  -o loop vyos-<version>-amd64.iso rootfs/
```

3. Create directory and unpack squashfs filesystem

```
root@deb10# mkdir unsquashfs
root@deb10# unsquashfs -f -d unsquashfs/ rootfs/live/filesystem.squashfs
```

4. Fix locales for correct system configuration loading

```
root@deb10# sed -i 's/^LANG=.*$/LANG=C.UTF-8/' unsquashfs/etc/default/locale
```

5. Optional step: Decrease docker image size by deleting not necessary files for container

```
root@deb10# rm -rf unsquashfs/boot/*.img
root@deb10# rm -rf unsquashfs/boot/*vyos*
root@deb10# rm -rf unsquashfs/boot/vmlinuz
root@deb10# rm -rf unsquashfs/lib/firmware/
root@deb10# rm -rf unsquashfs/usr/lib/x86_64-linux-gnu/libwireshark.so*
root@deb10# rm -rf unsquashfs/lib/modules/*amd64-vyos
```

6. Create a symbolic link to the configuration

```
root@deb10# ln -s /opt/vyatta/etc/config unsquashfs/config
```

7. Create docker image

```
root@deb10# tar -C unsquashfs -c . | docker import - vyos:<version> --change 'CMD ["/sbin/init"]'
```

8. Check that the image has been added to the Docker

```
root@deb10# docker image ls
```

```
REPOSITORY    TAG          IMAGE ID     CREATED          SIZE
vyos          <version>    <imageid>    N minutes ago    NMB
```

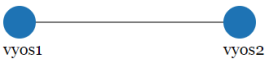
Deploying VyOS in Containerlab

Creating basic lab example

For this lab, we're using 2 VyOS nodes connected with eth1 interfaces

ContainerLab Topology 'vyosclab01'

Export as PNG



name	image	kind	group	state	ipv4 address	ipv6 address
vyos1	vyos:1.2.6	linux		running/Up 16 minutes	172.20.20.2/24	2001:172:20:20::2/64
vyos2	vyos:1.2.6	linux		running/Up 16 minutes	172.20.20.3/24	2001:172:20:20::3/64

Create a new lab folder and topology file

```
root@deb10# mkdir clab_vyos
root@deb10# cd clab_vyos
root@deb10# nano vyos_lab1.clab.yml

name: vyosclab01

topology:
  nodes:
    vyos1:
      kind: linux
      image: vyos:1.3.0
      cmd: /sbin/init
      binds:
        - /lib/modules:/lib/modules
    vyos2:
      kind: linux
      image: vyos:1.3.0
```

```
    cmd: /sbin/init
    binds:
      - /lib/modules:/lib/modules

  links:
    - endpoints: ["vyos1:eth1", "vyos2:eth1"]
```

Notice: ***cmd*** and ***binds*** fields used in the example above are mandatory in order to allow clab to correctly deploy VyOS nodes. Check if those values are defined correctly in case if you see the following error during the lab startup:

ERRO[0000] failed deploy phase for node "<node>": Error response from daemon: No command specified

Save the topology file and close it.

Notice: eth0 is reserved for management purposes and not available for the topology p2p links, however, it's shown in the interfaces status information, has automatically assigned management IP address and default route. It's recommended to avoid any configuration changes for eth0 to prevent unexpected issues in the lab.

Running the basic lab

Start the lab

```
root@deb10# clab deploy --topo  vyos_lab1.clab.yml
```

You will see the lab deployment process with results and nodes information

```
INFO[0000] Parsing & checking topology file: vyos_lab1.clab.yml
INFO[0000] Creating lab directory: /root/clab/clab_vyos_1.3.0/clab-vyosclab01
INFO[0000] Creating docker network: Name='clab', IPv4Subnet='172.20.20.0/24',
IPv6Subnet='2001:172:20:20::/64', MTU='1500'
INFO[0000] Creating container: vyos1
INFO[0000] Creating container: vyos2
INFO[0009] Creating virtual wire: vyos1:eth1 <--> vyos2:eth1
INFO[0009] Adding containerlab host entries to /etc/hosts file
+---+---------------------+--------------+-----------+-------+---------+---------------+----------------------+
| # |        Name         | Container ID |   Image   | Kind  |  State  | IPv4 Address  |     IPv6 Address
|
+---+---------------------+--------------+-----------+-------+---------+---------------+----------------------+
| 1 | clab-vyosclab01-vyos1 | d31615622f9f | vyos:1.3.0 | linux | running | 172.20.20.2/24 |
2001:172:20:20::2/64 |
| 2 | clab-vyosclab01-vyos2 | 405e8485248f | vyos:1.3.0 | linux | running | 172.20.20.3/24 |
2001:172:20:20::3/64 |
+---+---------------------+--------------+-----------+-------+---------+---------------+----------------------+
```

Connect to the vyos1 node using Docker (ssh by default is disabled in VyOS but it can be preconfigured within the mounted configuration that will be described in the next section)

```
root@deb10# docker exec -it clab-vyosclab01-vyos1 su vyos
vyos@vyos:/$ configure
vyos@vyos# set interfaces ethernet eth1 address 10.1.2.1/30
[edit]
vyos@vyos# set interfaces ethernet eth1 description "to-vyos2-eth1"
[edit]
vyos@vyos# commit ; save ; exit
Saving configuration to '/config/config.boot'...
Done
exit

vyos@vyos:/$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface       IP Address                S/L   Description
----------      -------------             ----  ---------------
eth0            172.20.20.2/24            u/u
                2001:172:20:20::2/64
eth1            10.1.2.1/30               u/u   to-vyos2-eth1
lo              127.0.0.1/8               u/u
                ::1/128
```

Configure vyos2 and check connectivity between nodes

```
root@deb10# docker exec -it clab-vyosclab01-vyos2 su vyos
vyos@vyos:/$ configure
[edit]
vyos@vyos# set interfaces ethernet et
eth0  eth1
[edit]
vyos@vyos# set interfaces ethernet eth1 address 10.1.2.2/30
[edit]
vyos@vyos# set interfaces ethernet eth1 description "to-vyos1-eth1"
[edit]
vyos@vyos# commit ; save ; exit
Saving configuration to '/config/config.boot'...
Done
exit
vyos@vyos:/$ ping 10.1.2.1
PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data.
64 bytes from 10.1.2.1: icmp_seq=1 ttl=64 time=0.084 ms
64 bytes from 10.1.2.1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 10.1.2.1: icmp_seq=3 ttl=64 time=0.038 ms
```

```
64 bytes from 10.1.2.1: icmp_seq=4 ttl=64 time=0.067 ms
64 bytes from 10.1.2.1: icmp_seq=5 ttl=64 time=0.035 ms
^C
--- 10.1.2.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4092ms
rtt min/avg/max/mdev = 0.035/0.051/0.084/0.022 ms
```

Notice: If in VyOS appears IPv6-related errors, for example, it cannot assign an IPv6 for an interface, it is necessary to enable IPv6 support in Docker. This can be done, by editing ***/etc/docker/daemon.json***:

```
{
    "ipv6": true,
    "fixed-cidr-v6": "fe80::/64"
}
```

Saving persistent configuration for the labs

As the Containerlab is using the Docker engine in order to create and manage devices, the configuration is not being saved after the lab stopped and containers are destroyed. To save the configuration and make it persistent on the next device boot, you can mount the configuration file or the whole VyOS /config directory with other files including encryption keys, etc.

There are a few specific points about VyOS and saving configuration persistently:

1. When you're mounting /config directory to the lab containers, there are 2 files always required to be included:

- ***config.boot*** - where configuration is stored and managed by the VyOS
- ***.vyatta_config*** - mandatory for every VyOS container

2. ***config.boot*** file must have special lines that are mandatory for the system to verify and identify the configuration file. These lines are unique and depend on the VyOS version. Here is an example for the VyOS 1.3.0:

```
vyos@vyos:~$ cat /config/config.boot
{
Configuration part has been removed for brevity
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version:
"wanloadbalance@3:ntp@1:webgui@1:dhcp-server@5:pptp@1:webproxy@2:quagga@6:qos@
```

```
1:firewall@5:ssh@1:dhcp-relay@2:config-management@1:l2tp@1:dns-forwarding@1:ipsec@5:
cluster@1:system@9:conntrack-sync@1:conntrack@1:snmp@1:mdns@1:nat@4:webproxy@1:b
roadcast-relay@1:vrrp@2:zone-policy@1" === */
/* Release version: 1.3.0 */
```

In order to make your mounted configuration file recognized by the VyOS, put these lines at the end of the configuration file before mounting it to the container.

3. It's possible that the mounted /config directory will result in wrong permissions of the configuration related files and directories. In that case, you'll need to manually change them in order to allow the system to correctly operate and manage configuration. The correct permissions are **root:vyattacfg**.

In case if you're getting the following error during configuration commit and save process

```
Couldn't open /opt/vyatta/etc/config/archive/commits - Permission denied at
/opt/vyatta/share/perl5/Vyatta/ConfigMgmt.pm line 108.
run-parts: /etc/commit/post-hooks.d/01vyatta-commit-revs.pl exited with return code 13
```

Run these commands to fix it quickly

```
vyos@vyos:~$ sudo chown root:vyattacfg /opt/vyatta/etc/config/archive/commits
vyos@vyos:~$ sudo chmod 775 /opt/vyatta/etc/config/archive/commits
```

Here is an example of a topology file with mounted pre-defined configuration including custom hostname and enabled ssh service on VyOS:

1. Create separate directories on the clab host for each node that is defined, paste empty ***.vyatta_config*** and custom ***config.boot*** file

```
root@deb10# cd clab_vyos
root@deb10# mkdir vyos1/config vyos2/config
root@deb10# touch vyos1/config/.vyatta_config vyos2/config/.vyatta_config
root@deb10# touch vyos1/config/config.boot vyos2/config/config.boot
```

2. Edit custom configuration files (same configuration on both hosts with different hostnames used)

```
root@deb10# nano vyos1/config/config.boot
service {
    ssh {
        disable-host-validation
    }
}
system {
    config-management {
```

```
    commit-revisions 100
  }
  host-name vyos1
  login {
    user vyos {
      authentication {
        encrypted-password
$6$QxPS.uk6mfo$9QBSo8u1FkH16gMyAVhus6fU3LOzvLR9Z9.82m3tiHFAxTtIkhaZSWssSgzt4v4dGA
L8rhVQxTg0oAG9/q11h/
        plaintext-password ""
      }
      level admin
    }
  }
  time-zone UTC
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version:
"wanloadbalance@3:ntp@1:webgui@1:dhcp-server@5:pptp@1:webproxy@2:quagga@6:qos@
1:firewall@5:ssh@1:dhcp-relay@2:config-management@1:l2tp@1:dns-forwarding@1:ipsec@5:
cluster@1:system@9:conntrack-sync@1:conntrack@1:snmp@1:mdns@1:nat@4:webproxy@1:b
roadcast-relay@1:vrrp@2:zone-policy@1" === */
/* Release version: 1.3.0 */
```

Exit and save the file.

3. Mount custom /config folder to the containers on the lab startup

```
root@deb10# nano vyos_lab1.clab.yml
name: vyosclab01

topology:
  nodes:
    vyos1:
      kind: linux
      image: vyos:1.3.0
      binds:
        - /lib/modules:/lib/modules
        - vyos1/config:/opt/vyatta/etc/config
    vyos2:
      kind: linux
      image: vyos:1.3.0
```

```
    binds:
      - /lib/modules:/lib/modules
      - vyos2/config:/opt/vyatta/etc/config

  links:
    - endpoints: ["vyos1:eth1", "vyos2:eth1"]
```

Exit and save the file.

4. Start the lab, wait for the nodes power-on and check if ssh works (as defined in the mounted custom configuration file)

```
root@deb10# clab deploy --topo vyos_lab1.clab.yml
INFO[0000] Parsing & checking topology file: vyos_lab1.clab.yml
INFO[0000] Creating lab directory: /root/clab/clab_vyos/clab-vyosclab01
INFO[0000] Creating docker network: Name='clab', IPv4Subnet='172.20.20.0/24',
IPv6Subnet='2001:172:20:20::/64', MTU='1500'
INFO[0000] Creating container: vyos1
INFO[0000] Creating container: vyos2
INFO[0002] Creating virtual wire: vyos1:eth1 <--> vyos2:eth1
INFO[0002] Adding containerlab host entries to /etc/hosts file
+---+----------------------+--------------+------------+-------+---------+---------------+----------------------+
| # |         Name         | Container ID |   Image    | Kind  |  State  |  IPv4 Address |     IPv6 Address
|
+---+----------------------+--------------+------------+-------+---------+---------------+----------------------+
| 1 | clab-vyosclab01-vyos1 | e38b80f7d851 | vyos:1.3.0 | linux | running | 172.20.20.3/24 |
2001:172:20:20::3/64 |
| 2 | clab-vyosclab01-vyos2 | 2ee00500e161 | vyos:1.3.0 | linux | running | 172.20.20.2/24 |
2001:172:20:20::2/64 |
+---+----------------------+--------------+------------+-------+---------+---------------+----------------------+

root@deb10# ssh vyos@clab-vyosclab01-vyos1
Welcome to VyOS
vyos@172.20.20.3's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
vyos@vyos1:~$
```

5. Make configuration changes on the node and save them

```
vyos@vyos1:~$ configure
[edit]
vyos@vyos1# set interfaces dummy dum0 address 10.0.0.1/32
[edit]
vyos@vyos1# commit ; save ; exit
Saving configuration to '/config/config.boot'...
Done
exit
vyos@vyos1:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                    S/L  Description
---------        ----------                    ---  -----------
dum0             10.0.0.1/32                   u/u
eth0             172.20.20.2/24                u/u
                 2001:172:20:20::2/64
eth1             -                             A/D
lo               127.0.0.1/8                   u/u
                 ::1/128
```

6. Stop the lab, wait until containers are destroyed and start it again

```
root@deb10# clab destroy --topo vyos_lab1.clab.yml
INFO[0000] Parsing & checking topology file: vyos_lab1.clab.yml
INFO[0000] Destroying lab: vyosclab01
INFO[0000] Removed container: clab-vyosclab01-vyos1
INFO[0000] Removed container: clab-vyosclab01-vyos2
INFO[0000] Removing containerlab host entries from /etc/hosts file

root@deb10# clab deploy --topo vyos_lab1.clab.yml
INFO[0000] Parsing & checking topology file: vyos_lab1.clab.yml
INFO[0000] Creating lab directory: /root/clab/clab_vyos/clab-vyosclab01
INFO[0000] Creating docker network: Name='clab', IPv4Subnet='172.20.20.0/24',
IPv6Subnet='2001:172:20:20::/64', MTU='1500'
INFO[0000] Creating container: vyos1
INFO[0000] Creating container: vyos2
INFO[0001] Creating virtual wire: vyos1:eth1 <--> vyos2:eth1
INFO[0001] Adding containerlab host entries to /etc/hosts file
+---+-----------------------+--------------+------------+-------+---------+----------------+----------------------+
| # |        Name           | Container ID |   Image    | Kind  |  State  |  IPv4 Address  |     IPv6 Address
|
+---+-----------------------+--------------+------------+-------+---------+----------------+----------------------+
| 1 | clab-vyosclab01-vyos1 | 0bc4066fe927 | vyos:1.3.0 | linux | running | 172.20.20.2/24 |
2001:172:20:20::2/64 |
```

```
| 2 | clab-vyosclab01-vyos2 | afa8760ff76b | vyos:1.3.0 | linux | running | 172.20.20.3/24 |
2001:172:20:20::3/64 |
+---+----------------------+--------------+------------+-------+---------+----------------+----------------------+
```

7. Connect to the vyos1 node and check that the changes are existing

```
root@deb10# ssh vyos@clab-vyosclab01-vyos1
Welcome to VyOS
vyos@clab-vyosclab01-vyos1's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

vyos@vyos1:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
dum0             10.0.0.1/32                       u/u
eth0             172.20.20.2/24                    u/u
                 2001:172:20:20::2/64
eth1             -                                 A/D
lo               127.0.0.1/8                       u/u
                 ::1/128
```

Connecting VyOS nodes to the clab bridge network

Sometimes it might be useful to connect lab nodes to the bridge network (testing VRRP for example) and provide plain L2 connectivity. Here are the steps to achieve this:

1. Create bridge network for the docker containers

```
root@deb10# ip link add br10 type bridge
root@deb10# ip link set up br10
root@deb10# ip link show br10
20: br10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9500 qdisc noqueue state UP mode
DEFAULT group default qlen 1000
    link/ether aa:c1:ab:48:ed:2f brd ff:ff:ff:ff:ff:ff
```

2. By default, traffic from containers connected to the default docker bridge network is not forwarded to the outside world. The same applies to the traffic between nodes that are connected to the Linux bridge with different interfaces. To solve both issues and enable forwarding, you need to change two settings

```
root@deb10# sysctl net.ipv4.conf.all.forwarding=1
root@deb10# sudo iptables -P FORWARD ACCEPT
```

Notice: These settings do not persist across a reboot, so you may need to add them to a start-up script.

3. Edit clab network topology file, add bridge node type and nodes connections with newly created bridge network

```
root@deb10# nano clab-01_vyos_vrrp
name: vyosclab01

topology:
  nodes:
    vyos1:
      kind: linux
      image: vyos:1.3.0
      binds:
        - /lib/modules:/lib/modules
        - vyos1/config:/opt/vyatta/etc/config
    vyos2:
      kind: linux
      image: vyos:1.3.0
      binds:
        - /lib/modules:/lib/modules
        - vyos2/config:/opt/vyatta/etc/config
    br10:
        kind: bridge

    links:
        - endpoints: ["vyos1:eth1", "br10:eth1"]
        - endpoints: ["vyos2:eth1", "br10:eth2"]
```

4. Start the lab and check connectivity

```
root@deb10# docker exec -it clab-01_vyos_vrrp-vyos1 su vyos
vyos@vyos1:/$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                    S/L  Description
---------        ----------                    ---  -----------
eth0             172.20.20.3/24                u/u
                 2001:172:20:20::3/64
```

```
eth1          10.10.10.1/24               u/u  to-br-clab
              10.10.10.254/24
lo            127.0.0.1/8                 u/u
              ::1/128

vyos@vyos1:/$ ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.054 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=0.055 ms
^C
--- 10.10.10.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.054/0.060/0.071/0.007 ms
vyos@vyos1:/$ show arp
Address              HWtype  HWaddress          Flags Mask          Iface
10.10.10.2           ether   aa:c1:ab:cf:70:e0  C                   eth1
172.20.20.1          ether   02:42:a1:d7:ce:ca  C                   eth0

vyos@vyos1:/$ show vrrp
Name    Interface     VRID  State    Last Transition
------  ----------  ------  -------  -----------------
LAN     eth1          10  MASTER   32m40s
```

Note that you can use another VyOS node as a bridge device as well, instead of using additional docker bridges.

Resources

Containerlab website
https://containerlab.srlinux.dev/

Docker website
https://www.docker.com/

VyOS Open source router and firewall platform website
https://vyos.io/

VyOS Containerlab Example
https://drive.google.com/file/d/1tK-d3RZjM3YKFUSIhgu3Sucf1KsKtlFY/view?usp=sharing

VyOS DockerHub repository
https://hub.docker.com/u/vyos