

# Network Architectures

OpenStack-Ansible supports a number of different network architectures, and can be deployed using a single network interface for non-production workloads or using multiple network interfaces or bonded interfaces for production workloads.

Our OpenStack-Ansible production environment reference architectures in this guide segments traffic using VLANs across multiple network interfaces or bonds.

The default networks used in an OpenStack-Ansible in this deployment can be observed in the following table:

Network	CIDR	VLAN
Management Network	172.29.236.0/22	10
Overlay Network	172.29.240.0/22	30
Storage Network	172.29.244.0/22	20

## Note

Unless, if we specifically do mention about a new network architecture we will be referring to the same network architecture above through out this document.

## Management Network

The `Management Network`, also referred to as the `Container network`, provides management of and communication between the infrastructure and OpenStack services running in containers or on metal. The management network uses a dedicated `VLAN` typically connected to the `br-mgmt` bridge, and may also be used as the primary interface used to interact with the server via `SSH`.

## Overlay Network

The `Overlay Network`, also referred to as the `Tunnel Network`, provides connectivity between hosts for the purpose of `tunneling` encapsulated traffic using `VXLAN`, `GRE`, or other protocols. The `Overlay Network` uses a dedicated `VLAN` typically connected to the `br-vxlan` bridge.

## Storage Network

The **Storage Network** provides segregated access to Block Storage from OpenStack services such as **Cinder**, **Glance**, **Ceph**, etc.. The **Storage Network** uses a dedicated **VLAN** typically connected to the **br-storage** bridge.

### Note

The **CIDRs** and **VLANs** listed for each network are examples and may be different in your environment.

Additional **VLANs** may be required for the following purposes:

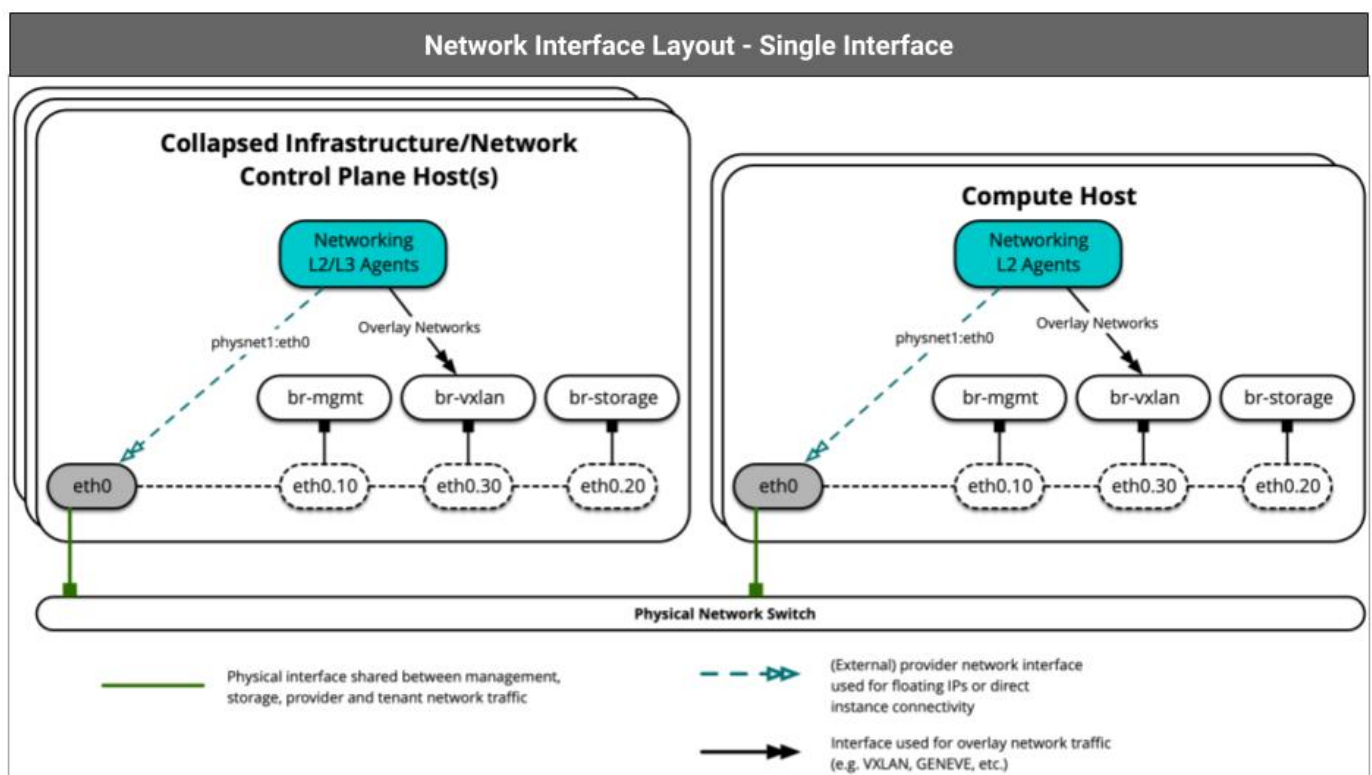
- External provider networks for **Floating IPs** and instances
- Self-service **Project/Tenant** networks for instances
- Other OpenStack services

## Network Interfaces

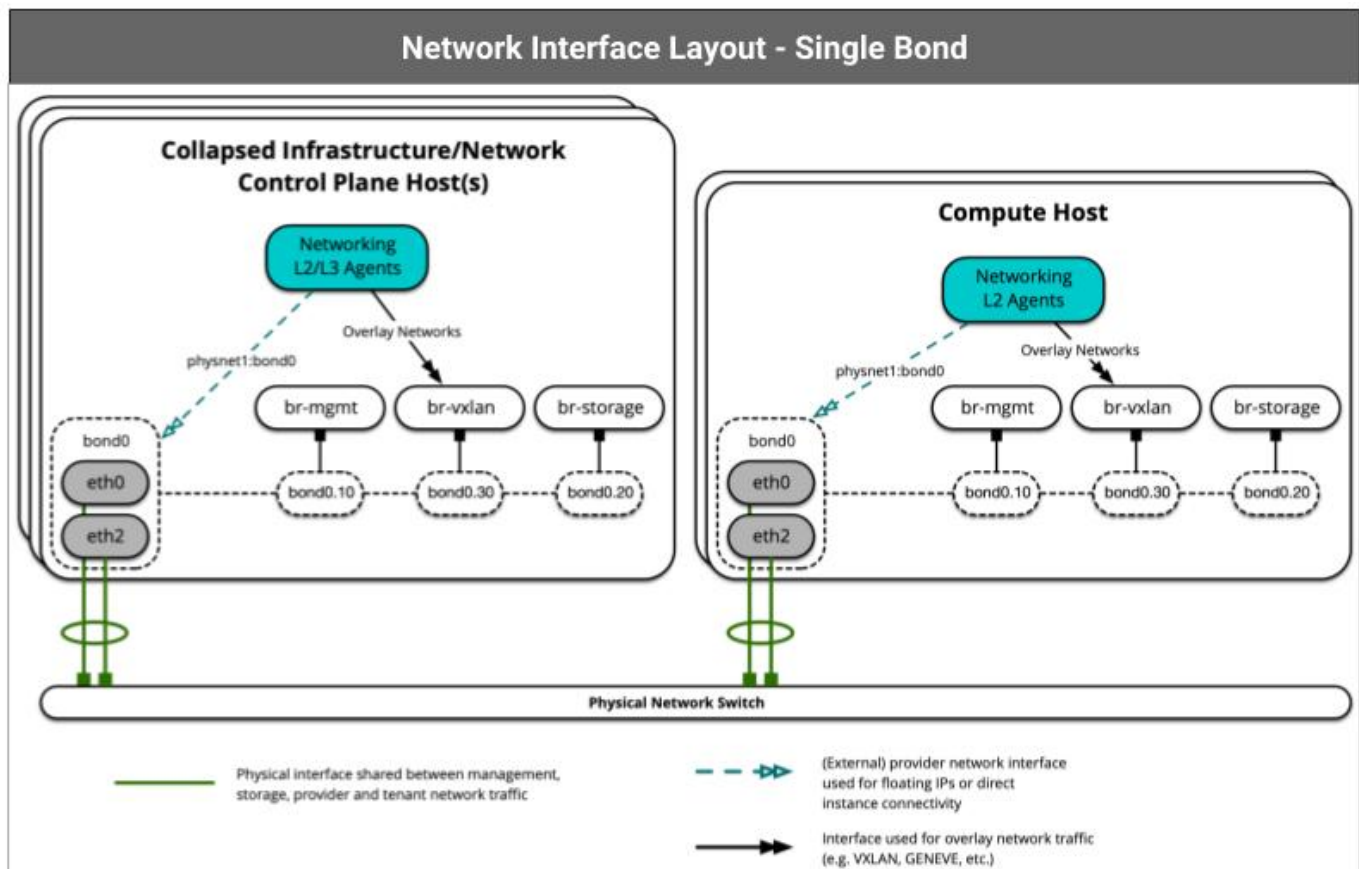
### Single Interface or Bond

OpenStack-Ansible supports the use of a **single** interface or set of **bonded** interfaces that carry **traffic** for OpenStack services as well as instances.

The following diagram demonstrates hosts using a single interface:



The following diagram demonstrates hosts using a single bond:



## Single Bond Sample Network Configuration

Each host will require the correct network bridges to be implemented.

The following is the `/etc/network/interfaces` file for `infra1` control plane host using a single bond in a Ubuntu host.

### Note

If your environment does not have `eth0`, but instead has `p1p1` or some other interface name, ensure that all references to `eth0` in all configuration files are replaced with the appropriate name. The same applies to additional network interfaces.

Please refer to [Configuring Network Interfaces](#) for more details.

### Important

This is a multi-NIC bonded configuration to implement the required bridges for OpenStack-Ansible. This illustrates the configuration of the first Infrastructure host `infra1` and the IP addresses assigned should be adapted for implementation on the other hosts. After implementing this configuration, the host will need to be rebooted.

Assuming that `eth0/1` and `eth2/3` are dual port NIC's we pair `eth0` with `eth2` for increased resiliency in the case of one interface card failing.

```

auto eth0

iface eth0 inet manual
    bond-master bond0
    bond-primary eth0

auto eth1
iface eth1 inet manual

auto eth2
iface eth2 inet manual
    bond-master bond0

auto eth3
iface eth3 inet manual

# Create a bonded interface. Note that the "bond-slaves" is set to none. This
# is because the bond-master has already been set in the raw interfaces for
# the new bond0.

auto bond0
iface bond0 inet manual
    bond-slaves none
    bond-mode active-backup
    bond-miimon 100
    bond-downdelay 200
    bond-updelay 200

# Container/Host management VLAN interface

auto bond0.10
iface bond0.10 inet manual
    vlan-raw-device bond0

# OpenStack Networking VXLAN (tunnel/overlay) VLAN interface

auto bond0.30
iface bond0.30 inet manual
    vlan-raw-device bond0

# Storage network VLAN interface (optional)

auto bond0.20
iface bond0.20 inet manual

# Container/Host management bridge

auto br-mgmt
iface br-mgmt inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports bond0.10

```

```

address 172.29.236.11
netmask 255.255.252.0
gateway 172.29.236.1
dns-nameservers 8.8.8.8 8.8.4.4

# OpenStack Networking VXLAN (tunnel/overlay) bridge
#
# Nodes hosting Neutron agents must have an IP address on this interface,
# including COMPUTE, NETWORK, and collapsed INFRA/NETWORK nodes.
#

auto br-vxlan
iface br-vxlan inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports bond0.30
    address 172.29.240.16
    netmask 255.255.252.0

# OpenStack Networking VLAN bridge
#
# The "br-vlan" bridge is no longer necessary for deployments unless Neutron
# agents are deployed in a container. Instead, a direct interface such as
# bond0 can be specified via the "host_bind_override" override when defining
# provider networks.
#
#auto br-vlan
#iface br-vlan inet manual
#    bridge_stp off
#    bridge_waitport 0
#    bridge_fd 0
#    bridge_ports bond0
# compute1 Network VLAN bridge
#auto br-vlan
#iface br-vlan inet manual
#    bridge_stp off
#    bridge_waitport 0
#    bridge_fd 0
#
# Storage bridge (optional)
#
# Only the COMPUTE and STORAGE nodes must have an IP address
# on this bridge. When used by infrastructure nodes, the
# IP addresses are assigned to containers which use this
# bridge.
#
auto br-storage
iface br-storage inet manual
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports bond0.20

```

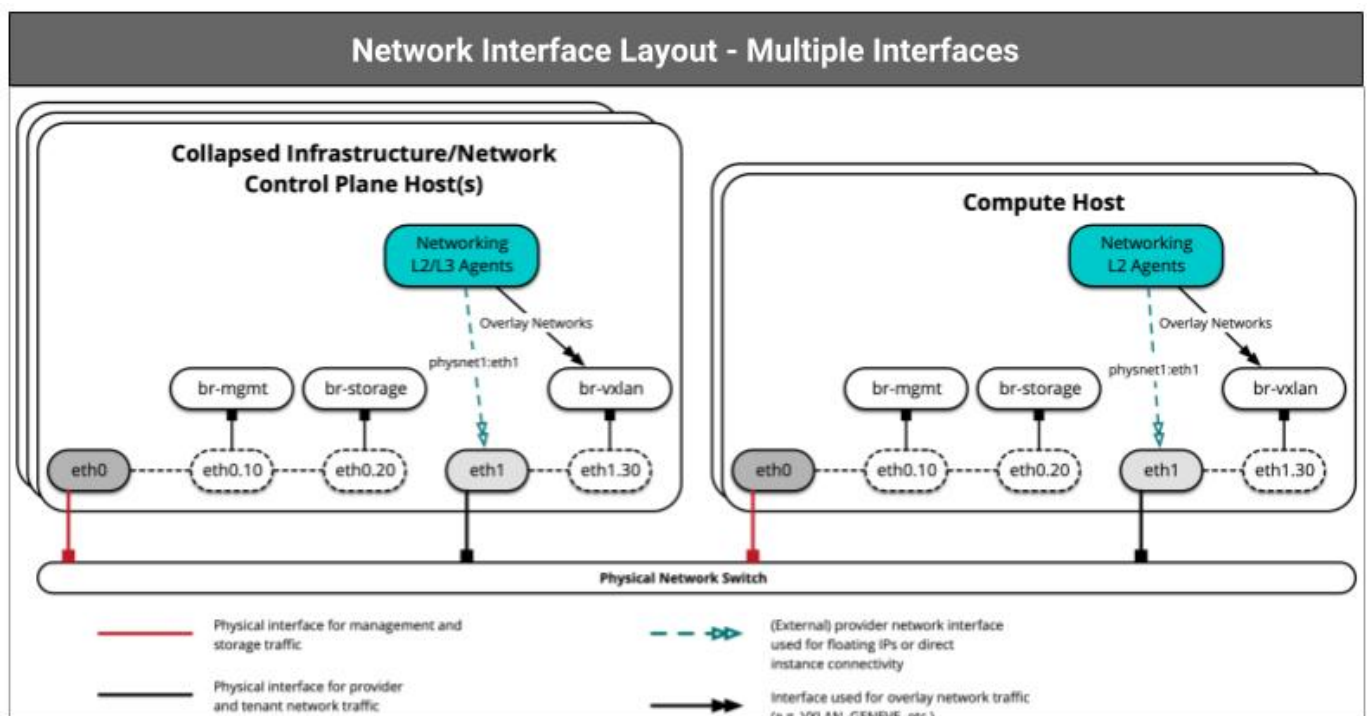
```
# compute1 Storage bridge

#auto br-storage
#iface br-storage inet static
#    bridge_stp off#auto br-storage
#    bridge_waitport 0
#    bridge_fd 0
#    bridge_ports bond0.20
#    address 172.29.244.16
#    netmask 255.255.252.0
```

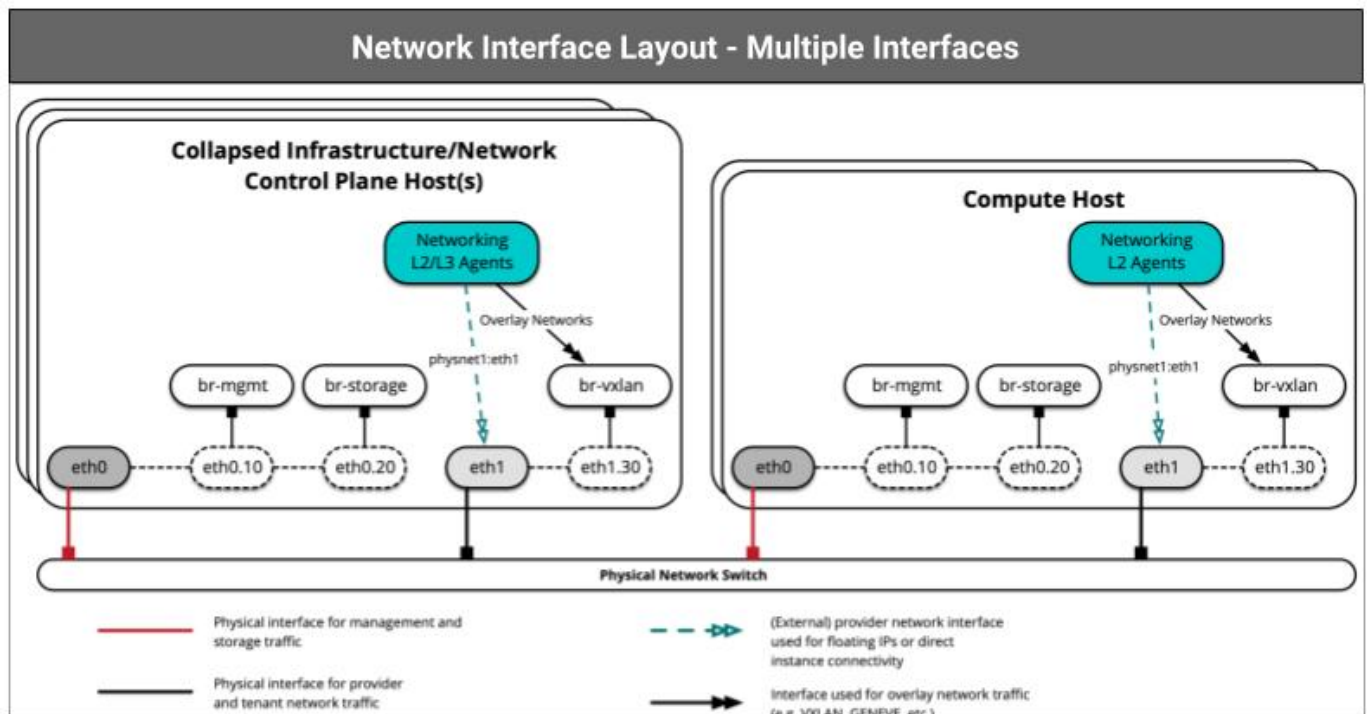
## Multiple Interfaces or Bonds

OpenStack-Ansible supports the use of a multiple interfaces or sets of bonded interfaces that carry traffic for OpenStack services and instances.

The following diagram demonstrates hosts using multiple interfaces:



The following diagram demonstrates hosts using multiple bonds:



## Multiples Bond Sample Network Configuration

### Note

If your environment does not have `eth0`, but instead has `p1p1` or some other interface name, ensure that all references to `eth0` in all configuration files are replaced with the appropriate name. The same applies to additional network interfaces.

Please refer to [Configuring Network Interfaces](#) for more details.

### Important

This is a multi-NIC bonded configuration to implement the required bridges for OpenStack-Ansible. This illustrates the configuration of the first Infrastructure host `infra1` and the IP addresses assigned should be adapted for implementation on the other hosts. After implementing this configuration, the host will need to be rebooted.

Assuming that `eth0/1` and `eth2/3` are dual port NIC's we pair `eth0` with `eth2` for increased resiliency in the case of one interface card failing.

```
auto eth0
iface eth0 inet manual
    bond-master bond0
    bond-primary eth0

auto eth1
iface eth1 inet manual
    bond-master bond1
    bond-primary eth1
```

```

auto eth2
iface eth2 inet manual
    bond-master bond0

auto eth3
iface eth3 inet manual
    bond-master bond1

# Create a bonded interface. Note that the "bond-slaves" is set to none. This
# is because the bond-master has already been set in the raw interfaces for
# the new bond0.
auto bond0
iface bond0 inet manual
    bond-slaves none
    bond-mode active-backup
    bond-miimon 100
    bond-downdelay 200
    bond-updelay 200

# This bond will carry VLAN and VXLAN traffic to ensure isolation from
# control plane traffic on bond0.
auto bond1
iface bond1 inet manual
    bond-slaves none
    bond-mode active-backup
    bond-miimon 100
    bond-downdelay 250
    bond-updelay 250

# Container/Host management VLAN interface
auto bond0.10
iface bond0.10 inet manual
    vlan-raw-device bond0

# OpenStack Networking VXLAN (tunnel/overlay) VLAN interface
auto bond1.30
iface bond1.30 inet manual
    vlan-raw-device bond1

# Storage network VLAN interface (optional)
auto bond0.20
iface bond0.20 inet manual
    vlan-raw-device bond0

# Container/Host management bridge
auto br-mgmt
iface br-mgmt inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports bond0.10
    address 172.29.236.11
    netmask 255.255.252.0
    gateway 172.29.236.1

```



```

dns-nameservers 8.8.8.8 8.8.4.4

# OpenStack Networking VXLAN (tunnel/overlay) bridge
#
# Nodes hosting Neutron agents must have an IP address on this interface,
# including COMPUTE, NETWORK, and collapsed INFRA/NETWORK nodes.
#

auto br-vxlan
iface br-vxlan inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports bond1.30
    address 172.29.240.16
    netmask 255.255.252.0

# OpenStack Networking VLAN bridge
#
# The "br-vlan" bridge is no longer necessary for deployments unless Neutron
# agents are deployed in a container. Instead, a direct interface such as
# bond1 can be specified via the "host_bind_override" override when defining
# provider networks.
#
#auto br-vlan
#iface br-vlan inet manual
#    bridge_stp off
#    bridge_waitport 0
#    bridge_fd 0
#    bridge_ports bond1

# compute1 Network VLAN bridge
#auto br-vlan
#iface br-vlan inet manual
#    bridge_stp off
#    bridge_waitport 0
#    bridge_fd 0
#

# Storage bridge (optional)
#
# Only the COMPUTE and STORAGE nodes must have an IP address
# on this bridge. When used by infrastructure nodes, the
# IP addresses are assigned to containers which use this
# bridge.
#
auto br-storage
iface br-storage inet manual
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports bond0.20

# compute1 Storage bridge

```

```
#auto br-storage
#iface br-storage inet static
#    bridge_stp off
#    bridge_waitport 0
#    bridge_fd 0
#    bridge_ports bond0.20
#    address 172.29.244.16
#    netmask 255.255.252.0
```