Red Hat OpenStack Platform 17.0

IPv6 Networking for the Overcloud

Configuring an overcloud to use IPv6 networking

Last Updated: 2024-03-21

Red Hat OpenStack Platform 17.0 IPv6 Networking for the Overcloud

Configuring an overcloud to use IPv6 networking

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide provides information on using the Red Hat OpenStack Platform director create an overcloud that uses IPv6 for endpoints. This includes information on how the director deploys an IPv6-based overcloud and the configuration options to achieve this.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. INTRODUCTION TO IPV6 FOR THE OVERCLOUD

Red Hat OpenStack Platform director creates a cloud environment called the overcloud. By default, the overcloud uses Internet Protocol version 4 (IPv4) to configure the service endpoints. However, the overcloud also supports Internet Protocol version 6 (IPv6) endpoints, which is useful for organizations that support IPv6 infrastructure.

This information is supplementary to the Director Installation and Usage guide. The same requirements specified in Director Installation and Usage also apply to this guide. Implement these requirements as necessary.

1.1. INTRODUCTION TO IPV6 NETWORKING

IPv6 is the latest version of the Internet Protocol standard. Internet Engineering Task Force (IETF) developed IPv6 as a means to combat the exhaustion of IP address from the current common IPv4 standard. IPv6 has various differences from IPv4 including:

Large IP Address Range

The IPv6 range is much larger than the IPv4 range.

Better End-to-End Connectivity

The larger IP range provides better end-to-end connectivity due to less reliance on network address translation.

No Broadcasting

IPv6 does not support traditional IP broadcasting. Instead, IPv6 uses multicasting to send packets to applicable hosts in a hierarchical manner.

Stateless Address Autoconfiguration (SLAAC)

IPv6 provides features for automatically configuring IP addresses and detecting duplicate addresses on a network. This reduces the reliance on a DHCP server to assign addresses.

IPv6 uses 128 bits (represented with 4 hexadecimals using groups of 16 bits) to define addresses while IPv4 uses only 32 bits (represented with decimal digits using groups of 8 bits). For example, a representation of an IPv4 address (192.168.0.1) looks like this:

Bits	Representation
11000000	192
10101000	168
00000000	0
00000001	1

For an IPv6 address (2001:db8:88ec:9fb3::1), the representation looks like this:

Bits	Representation
0010 0000 0000 0001	2001

Bits	Representation
0000 1101 1011 1000	0db8
1000 1000 1110 1100	88ec
1001 1111 1011 0011	9fb3
0000 0000 0000 0000	0000
0000 0000 0000 0000	0000
0000 0000 0000 0000	0000
0000 0000 0000 0001	0001

You can also represent IPv6 addresses without leading zeros in each bit group and omit a set of zero bit groups once for each IP address. In this example, you can represent the **0db8** bit grouping as just **db8** and omit the three sets of **0000** bit groups, which shortens the representation from **2001:0db8:88ec:9fb3:0000:0000:0000:0001** to **2001:db8:88ec:9fb3::1**. For more information, see "RFC 5952: A Recommendation for IPv6 Address Text Representation"

Subnetting in IPv6

Similar to IPv4, an IPv6 address uses a bit mask to define the address prefix as its network. For example, if you include a /**64** bit mask to the sample IP address **2001:db8:88ec:9fb3::1/64**, the bit mask acts as a prefix that defines the first 64 bits (**2001:db8:88ec:9fb3**) as the network. The remaining bits (**0000:0000:0000:0001**) define the host.

IPv6 also uses some special address types:

Loopback

The loopback device uses an IPv6 for the internal communication within the host. This device is always **::1/128**.

Link Local

A link local address is an IP address valid within a particular network segment. IPv6 requires each network device to have a link local address and use the prefix **fe80::/10**. However, most of the time, these addresses are prefixed with **fe80::/64**.

Unique local

A unique local address is intended for local communication. These addresses use a **fc00::/7** prefix.

Multicast

Hosts use multicast addresses to join multicast groups. These addresses use a **ff00::/8** prefix. For example, **FF02::1** is a multicast group for all nodes on the network and **FF02::2** is a multicast group for all routers.

Global Unicast

These addresses are usually reserved for public IP address. These addresses use a **2000::/3** prefix.

1.2. USING IPV6 IN RED HAT OPENSTACK PLATFORM

Red Hat OpenStack Platform (RHOSP) director maps OpenStack services to isolated networks. These networks include:

- Internal API
- Storage
- Storage Management
- Project(tenant) Networks (Neutron VLAN mode)
- External

For more information about these network traffic types, see the Director Installation and Usage guide.

Director also provides methods to use IPv6 communication for these networks. This means the required OpenStack services, databases, and other related services use IPv6 addresses to communicate. This also applies to environments that use a high availability solution involving multiple Controller nodes. This helps organizations integrate RHOSP with their IPv6 infrastructure.

Use the following table as a guide for which RHOSP networks support IPv6:

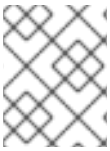
Network Type	Supported Internet Protocol (IP)	Notes
Internal API	• IPv6 • IPv4	
Storage	• IPv6 • IPv4	
Storage Management	• IPv6 • IPv4	
Project Networks	• Dual-stack (IPv4/v6) • IPv6 • IPv4	

Network Type	Supported Internet Protocol (IP)	Notes
Project Network Endpoints	• Dual stack (IPv4/v6) • IPv6 • IPv4	This refers to the IP address of the network hosting the project network tunnels, not the project networks themselves. IPv6 for network endpoints supports only VXLAN and Geneve. Generic routing encapsulation (GRE) is not yet supported.
External - Public API (and Horizon)	• IPv6 • IPv4	
External - Floating IPs	• Dual stack (IPv4/v6) • IPv4	IPv6 uses Global Unicast Addresses (GUAs) instead of NAT and floating IP addresses. The Networking (neutron) service expects the IPv6 addressing between project networks to use GUAs, with no overlap in GUAs across the project networks, and therefore can be routed without NAT. With dual stack (IPv4/v6), you can use floating IP addresses to only reach the IP addresses on IPv4 subnets.
Provider Networks	• Dual stack (IPv4/v6) • IPv6 • IPv4	IPv6 support is dependent on the project operating system.
Provisioning (PXE/DHCP)	• IPv6 • IPv4	

Network Type	Supported Internet Protocol (IP)	Notes
IPMI or other BMC	• IPv6 • IPv4	RHOSP communicates with baseboard management controller (BMC) interfaces over the Provisioning network. If BMC interfaces support dual stack IPv4 or IPv6, tools that are not part of RHOSP can use IPv6 to communicate with the BMCs.
Overcloud Provisioning network	IPv6	The Provisioning network used for ironic in the overcloud.
Overcloud Cleaning network	None	The isolated network used to clean a machine before it is ready for reuse.

Defining the scenario

The scenario for this guide is to create an overcloud with an isolated network that uses IPv6. Use heat templates and environment files to configure network isolation. This scenario also provides certain variants to these heat templates and environment files to demonstrate specific differences in configuration.



NOTE

In this scenario, the undercloud still uses IPv4 connectivity for PXE boot, introspection, deployment, and other services.

This guide uses a scenario similar to the advanced overcloud scenario. The main difference is the omission of the Ceph Storage nodes.

For more information about this scenario, see the Director Installation and Usage guide.

IMPORTANT

This guide uses the 2001:DB8::/32 IPv6 prefix for documentation purposes as defined in RFC 3849. Ensure that you substitute these example addresses for IPv6 addresses from your own network.

CHAPTER 2. CONFIGURING THE OVERCLOUD FOR IPV6

The following chapter provides the configuration required before running the **openstack overcloud deploy** command. This includes preparing nodes for provisioning, configuring an IPv6 address on the undercloud, and creating a network environment file to define the IPv6 parameters for the overcloud.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- Your network supports IPv6-native VLANs as well as IPv4-native VLANs.

2.1. CONFIGURING AN IPV6 ADDRESS ON THE UNDERCLOUD

The undercloud requires access to the overcloud Public API, which is on the External network. To accomplish this, the undercloud host requires an IPv6 address on the interface that connects to the External network.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- Your network supports IPv6-native VLANs as well as IPv4-native VLANs.
- An IPv6 address available to the undercloud.

Native VLAN or dedicated interface

If the undercloud uses a native VLAN or a dedicated interface attached to the External network, use the **ip** command to add an IPv6 address to the interface. In this example, the dedicated interface is **eth0**:

```
$ sudo ip link set dev eth0 up; sudo ip addr add 2001:db8::1/64 dev eth0
```

Trunked VLAN interface

If the undercloud uses a trunked VLAN on the same interface as the control plane bridge (**br-ctlplane**) to access the External network, create a new VLAN interface, attach it to the control plane, and add an IPv6 address to the VLAN. In this example, the External network VLAN ID is **100**:

```
$ sudo ovs-vsctl add-port br-ctlplane vlan100 tag=100 -- set interface vlan100 type=internal
$ sudo ip l set dev vlan100 up; sudo ip addr add 2001:db8::1/64 dev vlan100
```

Confirming the IPv6 address

Confirm the addition of the IPv6 address with the **ip** command:

```
$ ip addr
```

The IPv6 address appears on the chosen interface.

Setting a persistent IPv6 address

To make the IPv6 address permanent, modify or create the appropriate interface file in **/etc/sysconfig/network-scripts/**. In this example, include the following lines in either **ifcfg-eth0** or **ifcfg-vlan100**:

```
IPV6INIT=yes
IPV6ADDR=2001:db8::1/64
```

For more information, see How do I configure a network interface for IPv6? on the Red Hat Customer Portal.

2.2. REGISTERING AND INSPECTING NODES FOR IPV6 DEPLOYMENT

A node definition template (**instackenv.json**) is a JSON format file that contains the hardware and power management details for registering nodes. For example:

```
{
    "nodes":[
        {
            "mac":[
                "bb:bb:bb:bb:bb:bb"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.205"
        },
        {
            "mac":[
                "cc:cc:cc:cc:cc:cc"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.206"
        },
        {
            "mac":[
                "dd:dd:dd:dd:dd:dd"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.207"
```

```
        },
        {
            "mac":[
                "ee:ee:ee:ee:ee:ee"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.208"
        }
        {
            "mac":[
                "ff:ff:ff:ff:ff:ff"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.209"
        }
        {
            "mac":[
                "gg:gg:gg:gg:gg:gg"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.210"
        }
    ]
}
```

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- Nodes available for overcloud deployment.

Procedure

1. After you create the node definition template, save the file to the home directory of the stack user (/**home**/**stack**/**instackenv.json**), then import it into the director:

```
$ openstack overcloud node import ~/instackenv.json
```

This command imports the template and registers each node from the template into director.

2. Assign the kernel and ramdisk images to all nodes:

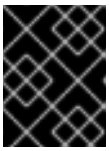
```
$ openstack overcloud node configure
```

The nodes are now registered and configured in director.

Verification steps

- After registering the nodes, inspect the hardware attribute of each node:

```
$ openstack overcloud node introspect --all-manageable
```



IMPORTANT

The nodes must be in the **manageable** state. Make sure this process runs to completion. This process usually takes 15 minutes for bare metal nodes.

2.3. TAGGING NODES FOR IPV6 DEPLOYMENT

After you register and inspect the hardware of your nodes, tag each node into a specific profile. These profile tags map your nodes to flavors, and in turn the flavors are assigned to a deployment role.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

Procedure

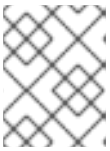
1. Retrieve a list of your nodes to identify their UUIDs:

```
$ ironic node-list
```

2. Add a profile option to the **properties/capabilities** parameter for each node. For example, to tag three nodes to use a controller profile and three nodes to use a compute profile, use the following commands:

```
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 5e3b2f50-fcd9-4404-b0a2-59d79924b38e add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 484587b2-b3b3-40d5-925b-a26a2fa3036f add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d010460b-38f2-4800-9cc4-d69f0d067efe add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d930e613-3e14-44b9-8240-4f3559801ea6 add
properties/capabilities='profile:compute,boot_option:local'
```

The addition of the **profile:control** and **profile:compute** options tag the nodes into each respective profile.



NOTE

As an alternative to manual tagging, use the automatic profile tagging to tag larger numbers of nodes based on benchmarking data.

2.4. CONFIGURING IPV6 NETWORKING

By default, the overcloud uses Internet Protocol version 4 (IPv4) to configure the service endpoints. However, the overcloud also supports Internet Protocol version 6 (IPv6) endpoints, which is useful for organizations that support IPv6 infrastructure. Director includes a set of environment files that you can use to create IPv6-based Overclouds.

For more information about configuring IPv6 in the Overcloud, see the dedicated IPv6 Networking for the Overcloud guide for full instructions.

2.4.1. Configuring composable IPv6 networking

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- Your network supports IPv6-native VLANs as well as IPv4-native VLANs.

Procedure

1. Copy the default **network_data** file:

```
$ cp /usr/share/openstack-tripleo-heat-templates/network_data.yaml /home/stack/.
```

2. Edit the local copy of the **network_data.yaml** file and modify the parameters to suit your IPv6 networking requirements. For example, the External network contains the following default network details:

```
- name: External
  vip: true
  name_lower: external
  vlan: 10
  ipv6: true
  ipv6_subnet: '2001:db8:fd00:1000::/64'
  ipv6_allocation_pools: [{'start': '2001:db8:fd00:1000::10', 'end':
'2001:db8:fd00:1000:ffff:ffff:ffff:fffe'}]
  gateway_ipv6: '2001:db8:fd00:1000::1'
```

   - **name** is the only mandatory value, however you can also use **name_lower** to normalize names for readability. For example, changing **InternalApi** to **internal_api**.
   - **vip: true** creates a virtual IP address (VIP) on the new network with the remaining parameters setting the defaults for the new network.
   - **ipv6** defines whether to enable IPv6.
   - **ipv6_subnet** and **ipv6_allocation_pools**, and **gateway_ip6** set the default IPv6 subnet and IP range for the network.

3. Include the custom **network_data** file with your deployment using the **-n** option. Without the **-n** option, the deployment command uses the default network details.

2.4.2. IPv6 network isolation in the overcloud

The overcloud assigns services to the provisioning network by default. However, director can divide overcloud network traffic into isolated networks. These networks are defined in a file that you include in the deployment command line, by default named **network_data.yaml**.

When services are listening on networks using IPv6 addresses, you must provide parameter defaults to indicate that the service is running on an IPv6 network. The network that each service runs on is defined by the file **network/service_net_map.yaml**, and can be overridden by declaring parameter defaults for individual **ServiceNetMap** entries. These services require the parameter default to be set in an environment file:

```
parameter_defaults:
  # Enable IPv6 for Ceph.
  CephIPv6: True
  # Enable IPv6 for Corosync. This is required when Corosync is using an IPv6 IP in the cluster.
  CorosyncIPv6: True
  # Enable IPv6 for MongoDB. This is required when MongoDB is using an IPv6 IP.
  MongoDbIPv6: True
  # Enable various IPv6 features in Nova.
  NovaIPv6: True
  # Enable IPv6 environment for RabbitMQ.
  RabbitIPv6: True
  # Enable IPv6 environment for Memcached.
  MemcachedIPv6: True
  # Enable IPv6 environment for MySQL.
  MysqlIPv6: True
  # Enable IPv6 environment for Manila
  ManilaIPv6: True
  # Enable IPv6 environment for Redis.
  RedisIPv6: True
```

The **environments/network-isolation.j2.yaml** file in the core heat templates is a Jinja2 file that defines all ports and VIPs for each IPv6 network in your composable network file. When rendered, it results in a **network-isolation.yaml** file in the same location with the full resource registry.

2.4.3. Configuring the IPv6 isolated network

The default heat template collection contains a Jinja2-based environment file for the default networking configuration. This file is **environments/network-environment.j2.yaml**. When rendered with our **network_data** file, it results in a standard YAML file called **network-environment.yaml**. Some parts of this file might require overrides.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- Your network supports IPv6-native VLANs as well as IPv4-native VLANs.

Procedure

- Create a custom environment file (**/home/stack/network-environment.yaml**) with the following details:

```
parameter_defaults:
  ControlPlaneDefaultRoute: 192.0.2.1
  ControlPlaneSubnetCidr: "24"
```

The **parameter_defaults** section contains the customization for certain services that remain on IPv4.

2.4.4. IPv6 network interface templates

The overcloud requires a set of network interface templates. Director contains a set of Jinja2-based Heat templates, which render based on your **network_data** file:

NIC directory	Description	Environment file
single-nic-vlans	Single NIC (**nic1**) with control plane and VLANs attached to default Open vSwitch bridge.	**environments/net-single-nic-with-vlans-v6.j2.yaml**
single-nic-linux-bridge-vlans	Single NIC (**nic1**) with control plane and VLANs attached to default Linux bridge.	**environments/net-single-nic-linux-bridge-with-vlans-v6.yaml**
bond-with-vlans	Control plane attached to **nic1**. Default Open vSwitch bridge with bonded NIC configuration (**nic2** and **nic3**) and VLANs attached.	**environments/net-bond-with-vlans-v6.yaml**
multiple-nics	Control plane attached to **nic1**. Assigns each sequential NIC to each network defined in the **network_data** file. By default, this is Storage to **nic2**, Storage Management to **nic3**, Internal API to **nic4**, Tenant to **nic5** on the **br-tenant** bridge, and External to **nic6** on the default Open vSwitch bridge.	**environments/net-multiple-nics-v6.yaml**

2.5. DEPLOYING AN IPV6 OVERCLOUD

To deploy an overcloud that uses IPv6 networking, you must include additional arguments in the deployment command.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.

Procedure

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml \
  -e /home/stack/templates/network-environment.yaml \
  --ntp-server pool.ntp.org \
  [ADDITIONAL OPTIONS]
```

The above command uses the following options:

- **--templates** - Creates the overcloud from the default heat template collection.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml** - Adds an additional environment file to the overcloud deployment. In this case, it is an environment file that initializes network isolation configuration for IPv6.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml** - Adds an additional environment file to the overcloud deployment. In this case, it is an environment file that initializes network isolation configuration for IPv6.
- **-e /home/stack/network-environment.yaml** - Adds an additional environment file to the overcloud deployment. In this case, it includes overrides related to IPv6.
  Ensure that the **network_data.yaml** file includes the setting **ipv6: true**. Previous versions of Red Hat OpenStack director included two routes: one for IPv6 on the External network (default) and one for IPv4 on the Control Plane. To use both default routes, ensure that the Controller definition in the **roles_data.yaml** file contains both networks in the **default_route_networks** parameter. For example, **default_route_networks: ['External', 'ControlPlane']**.
- **--ntp-server pool.ntp.org** - Sets the NTP server.

The overcloud creation process begins and director provisions the overcloud nodes. This process takes some time to complete. To view the status of the overcloud creation, open a separate terminal as the **stack** user and run:

```
$ source ~/stackrc
$ heat stack-list --show-nested
```

Accessing the overcloud

Director generates a script to configure and help authenticate interactions with your overcloud from the director host. The director saves this file (**overcloudrc**) in the home directory of the **stack** user. Run the following command to use this file:

```
$ source ~/overcloudrc
```

This loads the necessary environment variables to interact with your overcloud from the director host CLI. To return to interacting with the director host, run the following command:

```
$ source ~/stackrc
```

CHAPTER 3. POST-DEPLOYMENT IPV6 OPERATIONS

After you deploy the overcloud with IPv6 networking, you must perform some additional configuration.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.

3.1. CREATING AN IPV6 PROJECT NETWORK ON THE OVERCLOUD

The overcloud requires an IPv6-based Project network for instances. Source the **overcloudrc** file and create an initial Project network in **neutron**.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Your network supports IPv6-native VLANs as well as IPv4-native VLANs.

Procedure

1. Source the overcloud credentials file:

```
$ source ~/overcloudrc
```

2. Create a network and subnet:

```
$ openstack network create default --external --provider-physical-network datacentre --provider-network-type vlan --provider-segment 101

$ openstack subnet create default --subnet-range 2001:db8:fd00:6000::/64 --ipv6-address-mode slaac --ipv6-ra-mode slaac --ip-version 6 --network default
```

This creates a basic **neutron** network called **default**.

Verification steps

- Verify that the network was created successfully:

```
$ openstack network list
$ openstack subnet list
```

3.2. CREATING AN IPV6 PUBLIC NETWORK ON THE OVERCLOUD

After you configure the node interfaces to use the External network, you must create this network on the overcloud to enable network access.

Prerequisites

- A successful undercloud installation. For more information, see Installing director.
- A successful overcloud deployment. For more information, see Creating a basic overcloud with CLI tools.
- Your network supports IPv6-native VLANs as well as IPv4-native VLANs.

Procedure

1. Create an external network and subnet:

```
$ openstack network create public --external --provider-physical-network datacentre --provider-network-type vlan --provider-segment 100

$ openstack subnet create public --network public --subnet-range 2001:db8:0:2::/64 --ip-version 6 --gateway 2001:db8::1 --allocation-pool start=2001:db8:0:2::2,end=2001:db8:0:2::ffff --ipv6-address-mode slaac --ipv6-ra-mode slaac
```

   This command creates a network called **public** that provides an allocation pool of over 65000 IPv6 addresses for our instances.

2. Create a router to route instance traffic to the External network.

```
$ openstack router create public-router
$ openstack router set public-router --external-gateway public
```