BGP

Jem Camba - 2019-03-06 - in Networking

About Packet's BGP Support

Packet supports BGP (Border Gateway Protocol, read up more here) as a protocol for advertising routes to Packet servers in a local environment (called Local BGP), as well as to the greater Internet (called Global BGP).

Leveraging the powerful routing features of BGP will require that you operate a BGP speaking routing client, such as BIRD, Quagga or ExaBGP on your server. This routing client will control route advertisements via a BGP session to Packet's upstream routers. Packet will learn the routes that you are advertising (or not) and appropriately send traffic to your server(s).

Local BGP versus Global BGP

As mentioned above, Packet supports both Local and Global BGP options. In a Local BGP deployment, a customer uses an internal ASN (Autonomous System Number—more here) to control routes within a single Packet datacenter. This means that the routes are never advertised to the global Internet and the customer does not need to pay for or maintain a registered ASN with a Regional Internet Registry (RIR) like ARIN, APNIC or RIPE. The top use case for operating local BGP would be to perform failover or IP mobility between a collection of servers in a local datacenter.

Global BGP, on the other hand, requires a customer to have a registered ASN through one of the regional registries and advertise their own IP space.

Communities Supported

Packet supports the following BGP communities on routes learned from customers.

54825:666

- Function: BGP blackhole.
- Packet will blackhole traffic to your prefix, as well as signal supporting transit providers and peering partners to do the same.
- We support de-aggregation down to the /32 (IPv4) and /128 (IPv6) level with this community only.

54825:222

- Function: Anycast routing
- By tagging your routes with this community, Packet will advertise your routes to only transit and peering we maintain consistently on a global basis. Regional ISPs we connect with (e.g. Verizon in the New York metro area) will not learn your routes, as advertising to them may result in "scenic" routing for customers with global Anycast configurations.
- Please note that this community is not advised for normal use, as it will limit the number of available paths/providers you have access to. It should only be deployed by customers seeking BGP anycast topology, with multiple server instances deployed in each Packet datacenter.
- ***NOTE: This community is in beta testing at this time.***

Pricing

BGP requests are approved on a per project basis:

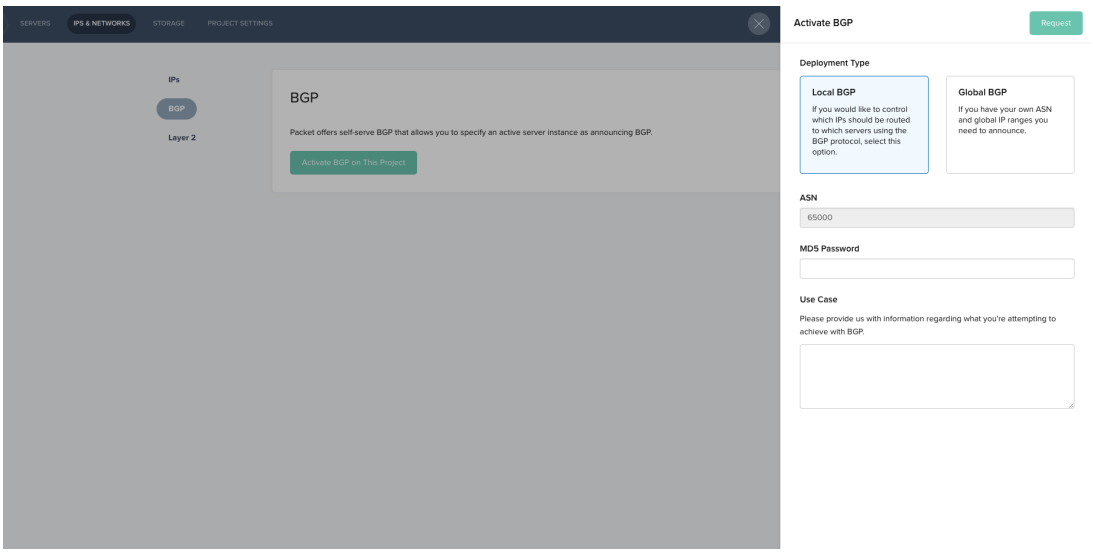
- Global BGP - Free (required on-boarding and review)
- Local BGP - Free

Getting Started

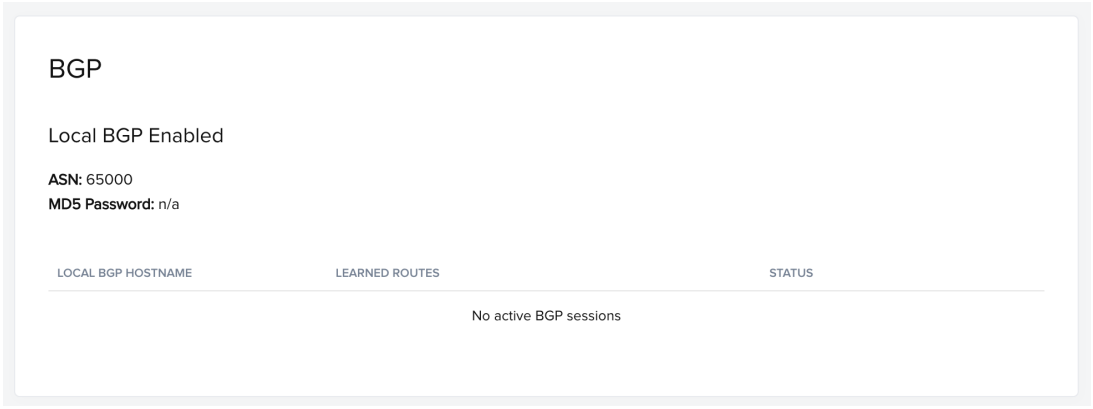
To get started, please create a Project in the Packet portal or navigate to an existing project that you'd like to enable for BGP support. Click on the IPs & Networks tab and navigate to the BGP request form (as shown below).

On the form, select either Global BGP or Local BGP. For Local BGP we default to using a private ASN of 65000. If you choose Global BGP, please enter your own public ASN. You may then enter an MD5 password for your BGP session. This is optional but highly recommended.

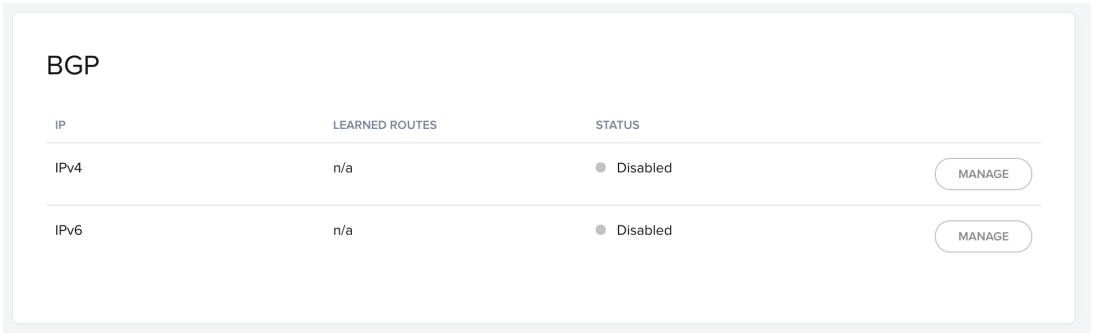
Lastly, please enter a few details about your use case and deployment so our engineers can quickly evaluate and approve your request.



Once you submit the request, Packet will review the details and get back to you shortly (usually in 24-48 hours). If we have any questions, we may contact you or we will advise you when BGP has been enabled for your project and you will then see your BGP details show up in the IPs & Networks tab, as follows:



Additionally, you will see some extra options on the server management page for any active devices in the specified project. You can choose to enable or disable IPv4 and IPv6 BGP for each server.



Note: Do not assign the subnet to any server, as that will create a static route thus causing an issue with the BGP setup.

Sample Local BGP Configuration

For this example, I will deploy a t1.small server, with Ubuntu on it, and then I will use BIRD to announce a public IP with BGP.

1. Configure the Elastic Ip that we will announce as a virtual loopback

Edit your interfaces file (*/etc/network/interfaces*) and add the following (where the *147.75.97.125* is the elastic IP):

```
auto lo:0
iface lo:0 inet static
  address 147.75.97.125
  netmask 255.255.255.255
```

Once you save that bring up the interface with **ifup lo:0**

Note: If you try to ping 147.75.97.125 (your elastic IP) from another computer you will see that it won't work. After we configure BGP, we will be able to ping it.

2. Installing BIRD

BIRD is an open source implementation for routing Internet Protocol packets on Unix-like operating systems. Later on in the configuration we will peer with the server's gateway, and we will exchange routes where we will announce our elastic Public IP.

```
apt-get install bird
```

3. Configuring IP forwarding

Depending on your operating system, you may need to adjust these commands.

For IPv4:

```
sysctl net.ipv4.ip_forward=1
```

For IPv6:

```
sysctl net.ipv6.conf.all.forwarding=1
```

4. Edit the bird configuration file

You will find the file that BIRD uses at */etc/bird/bird.conf*

I like to backup the existing file, which has some good explanations and examples.

```
mv /etc/bird/bird.conf /etc/bird/bird.conf.original
```

Then write a new configuration file for our setup, as follows:

```
nano /etc/bird/bird.conf
```

and add the following:

```
filter packetdns {
# IPs to announce (the elastic ip in our case)
# Doesn't have to be /32. Can be lower
if net = 147.75.79.170/32 then accept;
}
# your (Private) bond0 IP below here
router id 10.10.237.129;
protocol direct {
interface "lo"; # Restrict network interfaces it works with
}
protocol kernel {
# learn; # Learn all alien routes from the kernel
persist; # Don't remove routes on bird shutdown
scan time 20; # Scan kernel routing table every 20 seconds
import all; # Default is import all
export all; # Default is export none
# kernel table 5; # Kernel table to synchronize with (default: main)
}
# This pseudo-protocol watches all interface up/down events.
protocol device {
scan time 10; # Scan interfaces every 10 seconds
}
# your default gateway IP below here
protocol bgp {
export filter packetdns;
local as 65000;
neighbor 10.10.10.237.128 as 65530;
#password "md5password";
}
```

Save that and restart the bird service, as follows:

```
service bird restart
```

5. Enable BGP for the server in the portal via the server detail page:

Go to the BGP Tab of the Server's Detail Page, and click Manage for IPv4 or IPv6.

Here you can find a sample BGP config for the specific server, as it includes the right router ID and Neighbor IP. Click Enable.

Note! It takes up to 5-10 minutes for BGP to come up, and it will learn the route(s) that we are announcing from our server.

In the server we can check the status by opening the bird daemon with the birdc command. Then we check the status of the BGP announcement with:

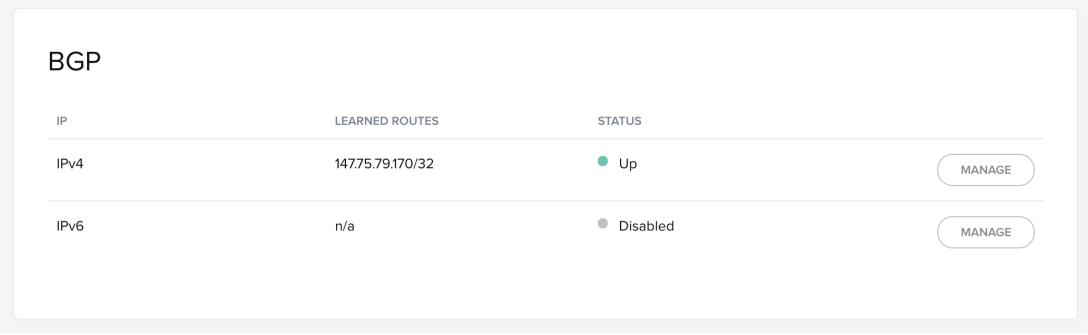
```
show protocols all bgp1
```

If the peering has been successful we will see it:

```
bird> show protocols all bgp1
name proto table state since info
bgp1 BGP master up 18:06:06 Established
 Preference: 100
 Input filter: ACCEPT
 Output filter: packetdns
 Routes: 0 imported, 1 exported, 0 preferred
 Route change stats: received rejected filtered ignored accepted
Import updates: 0 0 0 0 0
Import withdraws: 0 0 --- 0 0
Export updates: 1 0 0 --- 1
Export withdraws: 0 --- --- --- 0
 BGP state: Established
Neighbor address: 10.99.69.8
Neighbor AS: 65530
Neighbor ID: 192.80.8.235
Neighbor caps: refresh restart-aware AS4
Session: external AS4
Source address: 10.99.69.9
Hold timer: 85/90
Keepalive timer: 26/30
```

As you can see, the BGP state is Established and we are exporting 1 route.

If you check the server detail page, you will also see the route learned.



Now, finally, everything seems good and we should be able to ping that IP.

Congratulations! You are now announcing your IP to the Internet and controlling its route!

Tags

bgp