./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 01 - Lab Access

Prerequisites

>> Laptop with SSH client (PuTTy). The Windows client can be downloaded here.

Lab Assignment & Credentials

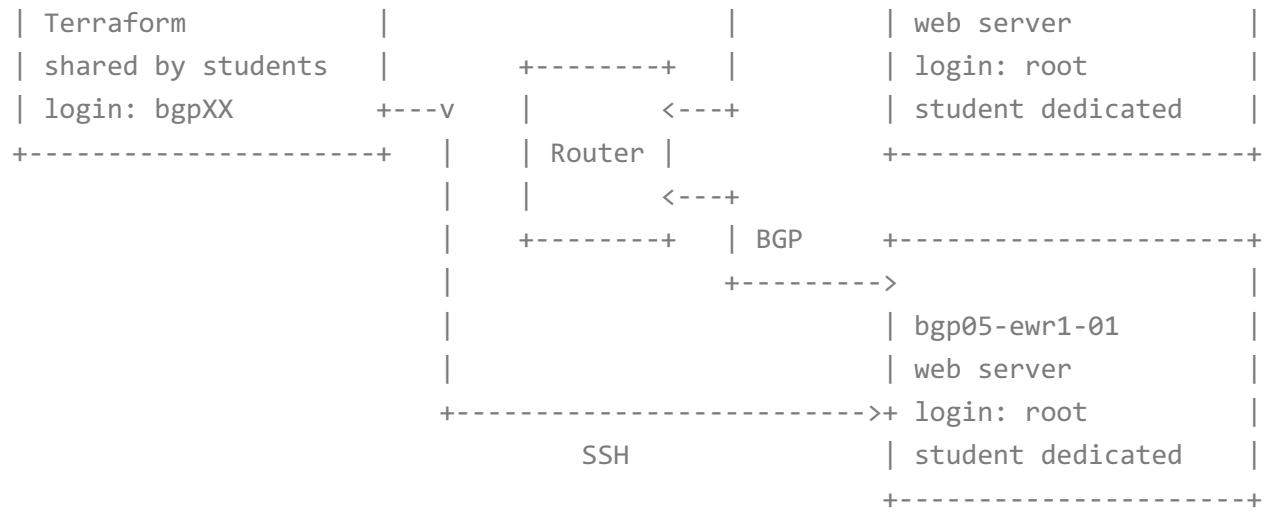
On the whiteboard/projector, there will be a link to an etherpad listing all the available lab environments along with the default password. Follow the link and write your name alongside a lab number (i.e. bgp03 - John Doe).

Take note of the name of the "lab master" server on the whiteboard/projector. This will be the jump server from where you will access

If you ever need a new lab environment, return to this page and simply assign yourself a new one. Mark any old/broken lab environments as "broken/recycle" and it will be rebuilt.

Deployment Layout

This lab consists of a "Lab Master" which is shared by all the students using the "bgpXX" login (replacing XX with lab assignment number, i.e. bgp05). Within that account directory is a deployed Terraform consisting of two physical servers (web servers). These hosts are dedicated to the students lab.



Lab Master Access

With your assigned lab username (i.e. bgp03), log into the lab master server using the your assigned lab and the password. You'll need to use a SSH client (i.e. PuTTy).

```
ssh <your_lab_username>@<lab_master_server>
```

Verify Deployed Web Instances

We've already taken the liberty of deploying a number of servers into your environment and setting them up with IPv6 and Anycast. This way you can see the end result. Don't worry, you'll get a chance to deploy it all yourself shortly.

Let's verify that all your hosts are deployed OK. You should have two physical hosts deployed, each with IPv4 and IPv6 addresses.

```
cd WorkspaceTemplate
terraform output


Anycast IPv6 Address = 2604:1380:2:2303::1
Anycast IPv6 Network = 2604:1380:2:2303::/64
IPv6 Anycast Curl Command = curl http://[2604:1380:2:2303::1]
SSH Access Server 0 = ssh root@147.75.98.155 -i mykey
Server IPs v4 = [
    147.75.98.155,
    147.75.76.169
]
Server IPs v6 = [
```

```
    2604:1380:2:2300::7,
    2604:1380:2:2300::d
 ]
```

And let's do a quick network connectivity check to each host.

```
ping -c 5 <Server #1 IP v4>
ping -c 5 <Server #2 IP v4>
```

You should see something like:

```
bgp03@stl:~/WorkspaceTemplate$ ping -c 5 147.75.194.9
PING 147.75.194.9 (147.75.194.9) 56(84) bytes of data.
64 bytes from 147.75.194.9: icmp_seq=1 ttl=60 time=0.206 ms
64 bytes from 147.75.194.9: icmp_seq=2 ttl=60 time=0.121 ms
64 bytes from 147.75.194.9: icmp_seq=3 ttl=60 time=0.183 ms
64 bytes from 147.75.194.9: icmp_seq=4 ttl=60 time=0.119 ms
64 bytes from 147.75.194.9: icmp_seq=5 ttl=60 time=0.176 ms

--- 147.75.194.9 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4093ms
rtt min/avg/max/mdev = 0.119/0.161/0.206/0.034 ms
bgp03@stl:~/WorkspaceTemplate$ ping -c 5 147.75.195.57
PING 147.75.195.57 (147.75.195.57) 56(84) bytes of data.
64 bytes from 147.75.195.57: icmp_seq=1 ttl=60 time=0.169 ms
64 bytes from 147.75.195.57: icmp_seq=2 ttl=60 time=0.111 ms
64 bytes from 147.75.195.57: icmp_seq=3 ttl=60 time=0.169 ms
64 bytes from 147.75.195.57: icmp_seq=4 ttl=60 time=0.156 ms
64 bytes from 147.75.195.57: icmp_seq=5 ttl=60 time=0.165 ms

--- 147.75.195.57 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4073ms
rtt min/avg/max/mdev = 0.111/0.154/0.169/0.022 ms
```

All the hosts should reply back from the ping requests. If, for some reason, your hosts are not responding, please go back and pick a different lab assignment and reverify.

Next Steps

Once you've validated your environment, proceed to Lab 2

./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 02 - Verifying Anycast

Prerequisites

>> Access to the lab master server from Lab 01
>> Validate that all web server instances are running OK from Lab 01

Overview

Each of the deployed hosts is running a web server that simply returns back the hostname. We're going to validate that the web server on each is running and returning a different hostname. We're then going to check the load balancing using the anycast address to show that traffic is distributed across all the hosts.

Verify Individual Web Servers

Each host running a web server that returns back it's hostname. Let's verify that each host web server is running correctly and that the correct hostname is being returned.

Get the list of hosts again

```
terraform output
```

Verify IPv4

```
curl http://<Server IP v4 #1>
curl http://<Server IP v4 #2>
```

Our sample output below shows the output from the web server as "bgp03-ewr1-00". The hostname consists of the lab number (bgp03), data center (ewr01), and host instance (00). Each lab initially has two host instances (00 and 01).

```
bgp03@stl:~/WorkspaceTemplate$ curl http://147.75.98.155/
bgp03-ewr1-00
```

Let's do the same check with IPv6. Make sure to use the square brackets around the IPv6 address.

```
curl http://[<Server IPs v6 #1>]
curl http://[<Server IPs v6 #2>]
```

Verify Anycast Capabilities

We've verified that each individual server is running so now validate that a traffic is split across all hosts using the anycast address. Repeat the command several times to validate that traffic is split across all the hosts.

```
terraform output
curl http://[Anycast IPv6 Address]/
curl http://[Anycast IPv6 Address]/
curl http://[Anycast IPv6 Address]/
```

In this sample output, traffic ends up at host bgp03-ewr1-00 and bgp03-ewr1-01.

```
bgp03@stl:~/WorkspaceTemplate$ curl http://[2604:1380:2:2303::1]
bgp03-ewr1-00
bgp03@stl:~/WorkspaceTemplate$ curl http://[2604:1380:2:2303::1]
bgp03-ewr1-01
bgp03@stl:~/WorkspaceTemplate$ curl http://[2604:1380:2:2303::1]
bgp03-ewr1-01
```

Next Steps

Once you're done and have verified that Anycast load balancing is working, proceed to Lab 3

./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 03 - Examine IPv6 and BGP Settings

Goals

Understand the network and BGP routing configuration.

Prerequisites

>> Access to the running host instances

Log into a Web Server via SSH

From the lab master, log into the deployed host instances using the installed SSH key. As part of the lab deployment, al the deployed hosts were embedded with an SSH key (my_key). The full SSH command is listed as “SSH Access Server 0” when `terraform output` is run.

```
terraform output
ssh root@<Server IP v4> -i my_key
```

Examine the v6 Addressing

On the deployed host, display and examine the IPv6 addresses assigned to the host.

```
ip -6 a show
```

Sample output (your IPv6 addresses will differ):

```
root@bgp01-ewr1-00:~# ip -6 a show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN qlen 1000
    inet6 2604:1380:2:2301::1/64 scope global deprecated
       valid_lft forever preferred_lft 0sec
```

```
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2604:1380:2:2300::3/127 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::ec4:7aff:fee5:484c/64 scope link
       valid_lft forever preferred_lft forever
```

Take note of the ::1/64 address on the lo interface. That same address has been assigned to all the hosts in the cluster and communicated to the upstream router via BGP. It's the address used for the anycast failover.

The subnet assigned, :2301:, is assigned to this lab. The whole space 2604:1380:2:2301::/64 is available for your use as you wish. Only the ::1 address has been used so far.

The IPv6 address to the bond0 interface is a point to point link assigned to the host when it boots.

Examine Network Interface File

These network settings are defined in the network interfaces file. We'll take a look later how this file is automatically generated. For now, take a look at the format and the values set within. Take a look at the last few lines where the anycast ipv6 address is assigned to lo:0.

```
more /etc/network/interfaces
```

Examine BGP Data

Next we're going to examine the running BGP routing details using BIRD, an open source, software router.

Let's see what routing protocols are running. (Make sure to add the 6 to run the IPv6 version of birdc.)

```
birdc6 show protocols
```

You'll see that BGP is running.

Now examine the full BGP routing details.

```
birdc6 show protocols all bgp1
```

Sample output:

```
root@bgp03-ewr1-00:~# birdc6 show protocols all bgp1
BIRD 1.6.3 ready.
name     proto    table    state  since       info
bgp1     BGP      master   up     17:49:21    Established
  Preference:     100
  Input filter:   ACCEPT
  Output filter:  packet_bgp
  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: 2604:1380:2:2300::6
    Neighbor AS:      65530
    Neighbor ID:      147.75.36.66
    Neighbor caps:    refresh restart-aware AS4
    Session:          external AS4
    Source address:   2604:1380:2:2300::7
    Hold timer:       84/90
    Keepalive timer:  11/30
```

Examine BIRD Configuration file

The BIRD software processes using a configuration file that details what routing protocols to run (BGP in our case)

```
more /etc/bird/bird6.conf
```

Take note of the “filter packet_bgp” and the network that is being communicated via BGP to the neighbor router. As you can see, this is the /64 that has been assigned to this lab.

Next Steps

Once you’ve verified the BGP information proceed to Lab 4

./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 04 - Examine Webserver & BGP Config Generation

Goals

>> Understand how the configuration files are generated

Prerequisites

>> Access to the lab master server from Lab 01

Lab Master Access

With your assigned lab username (i.e. bgp03), log into the lab master server using the your assigned lab and the password. You'll need to use a SSH client (i.e. PuTTy).

```
ssh <your_lab_username>@<lab_master_server>
```

Examine Web Terraform Config

Let's take a look how the webserver is configured to display the hostname.

```
cd WorkspaceTemplate/
more apache.tf
```

Very simply, apache, a web server is installed, and the default webpage (index.html) is written out to simply contain the hostname of the host.

Generating the IPv6 Subnets

The setting of the IPv6 anycast address is a little more complex but straight forward. Let's take a look at the Terraform configuration that

handles the network allocation.

```
more ipv6.tf
```

The `cidrsubnet` Terraform command is the

```
anycast_network    = "${cidrsubnet(data.packet_precreated_ip_block.ipv6.cidr_notatio
```

The “packet_precreated_ip_block” is the IPv6 block that has been assigned to this project. The `cidrsubnet` Terraform command then takes the large IPv6 block and splits it out into 2^8 subnets (the 8 represents the number of additional subnet bits).

This results in 256 subnets available across all the labs. Using the lab number, one of the individual subnets it then assigned to the lab.

```
anycast_addr      = "${cidrhost(local.anycast_network,1)}"
```

Within this subnet, the first (1) IP address is then assigned as the anycast address used for across all the hosts.

Configuring the host networking via Terraform

Using a Terraform template file, the networking information is used to update a template which is then run on the host. This sets up the IPv6 networking and BIRD BGP configuration file.

```
more templates/enable_bgp.tpl
```

The last few lines show how the network interfaces file is appended with the anycast IPv6 address.

Examine BGP Terraform Config

BIRD, the BGP software router, is installed using Terraform in bird.tf.

```
more bird.tf
```

The IPv6 networking and BIRD template file is configured via `configure_bird.tf`. This Terraform file defines all of the variables that are used in the template.

```
more configure_bird.tf
```

Next Steps

Once you feel comfortable on how Terraform is used to configure the IPv6 and BGP routing, proceed to Lab 5

./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 05 - Growing the cluster

Goals

>> Understand how to grow the cluster to include additional hosts that are automatically assigned to the anycast and have traffic balanced across.

Prerequisites

>> Access to the lab master server from Lab 01

Lab Master Access

With your assigned lab username (i.e. bgp03), log into the lab master server using the your assigned lab and the password. You'll need to use a SSH client (i.e. PuTTy).

```
ssh <your_lab_username>@<lab_master_server>
```

Increate the host count

The number of hosts is defined in the Terraform variables configuration file `vars.tf`. Increase it from the current count of 2 to 3.

```
cd WorkspaceTemplate/
vi vars.tf
```

The sample below already has the count increased to 3.

```
# number of hosts to deploy
variable "instance_count" {
    default = "3"
}
```

Examine Terraform Plan

Before applying this change to the network, let's examine what Terraform is planning to do. Items marked as [2] refer to our third instance which is to be deployed. Instances [0] and [1] have already been deployed.

```
terraform plan | more
```

In brief, Terraform will be:

```
>> packet_device.hosts[2] - deploying the bare metal host [2]
>> apache[2] - configuring apache on host [2]
>> bird[2] - installing BIRD on host [2]
>> configure_bird[2] - configuring BIRD on host [2]
>> enable_bgp_device_session[2] - notifying the upstream router that host
   [2] is an active BGP neighbor
```

Execute the Terraform Plan

Confident that Terraform will be doing the right thing, go ahead and apply the Terraform plan.

```
terraform apply --auto-approve
```

It'll take a approximately 3 minutes for the bare metal host to come online and then a few more minutes for the software installation.

Validate the new Host and Anycast

Terraform will now report back three hosts and the new host will be broadcasting BGP to the upstream router. Let's verify it is all functioning correctly.

```
terraform output
curl http://<Server IP v4 #1>
curl http://<Server IP v4 #2>
curl http://<Server IP v4 #3>
```

Now test the anycast address and validate that the new server is in the rotation. It might take a minute for BGP to sync and it to appear.

```
curl http://[Anycast IPv6 Address]
```

Repeat the `curl` command until you see the new web server show up.

In our sample below, the new server (02) it appeared on the second curl.

```
bgp03@stl:~/WorkspaceTemplate$  curl http://[2604:1380:2:2303::1]
bgp03-ewr1-00
bgp03@stl:~/WorkspaceTemplate$  curl http://[2604:1380:2:2303::1]
bgp03-ewr1-02
bgp03@stl:~/WorkspaceTemplate$  curl http://[2604:1380:2:2303::1]
bgp03-ewr1-01
```

Next Steps

Once you've validate that the bonus server is functioning and part of the anycast, proceed to Lab 6

./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 06 - Shrinking the Network

Goals

>> Examine how to shrink the network

Prerequisites

>> Access to the lab master server from Lab 01

Lab Master Access

With your assigned lab username (i.e. bgp03), log into the lab master server using the your assigned lab and the password. You'll need to use a SSH client (i.e. PuTTy).

```
ssh <your_lab_username>@<lab_master_server>
```

Decrement the host count

The number of hosts is defined in the Terraform variables configuration file `vars.tf`. Decreate it from the current count to 1.

```
cd WorkspaceTemplate/
vi vars.tf
```

The sample below already has the count decreased to 1.

```
# number of hosts to deploy
variable "instance_count" {
    default = "1"
}
```

Examine Terraform Plan

Before applying this change to the network, let's examine what Terraform is planning to do. Most notably, we see that the physical hosts are going to be released/destroyed.

```
terraform plan | more
```

Execute the Terraform Plan

Confident that Terraform will be doing the right thing, go ahead and apply the Terraform plan.

```
terraform apply --auto-approve
```

Destroying hosts is fast and Terraform will complete quickly.

Validate the new Host and Anycast

Terraform will now report back a single hosts. Let's verify it is all functioning correctly.

```
terraform output
curl http://<Server IP v4 #1>
```

Now test the anycast address to make sure that just the single host is returned.

```
curl http://[Anycast IPv6 Address]
```

Repeat the `curl` command to see that just the single remaining host is returned.

Next Steps

Once you've released the hardware, proceed to Lab 7

./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 07 - Adding an additional Anycast

Goal

>> Learn to modify Terraform
>> Add an additional anycast IPv6 address

Prerequisites

>> Access to the lab master server from Lab 01

Lab Master Access

With your assigned lab username (i.e. bgp03), log into the lab master server using the your assigned lab and the password. You'll need to use a SSH client (i.e. PuTTy).

```
ssh <your_lab_username>@<lab_master_server>
```

Define the second Anycast address

The first (:1) IPv6 in the subnet is already in use so we'll create a second IPv6 address and use :2

```
vi ipv6.tf
```

Call the new address "anycast_addr_2" and assign it the second address in the subnet.

```
locals {
  # 8 bits is 2^8 or 256 labs
  anycast_network    = "${cidrsubnet(data.packet_precreated_ip_block.ipv6.cidr_notat
  anycast_addr       = "${cidrhost(local.anycast_network,1)}"
```

```
  anycast_addr_2    = "${cidrhost(local.anycast_network,2)}"
}
```

Output the new Anycast address

The “output.tf” displays the generate values. Update it so that this new address is dislayed.

```
output "Anycast IPv6 Address 2" {
  value = "${local.anycast_addr_2}"
}
```

Execute the Terraform Plan

Executing the Terraform plan will output the new Anycast address.

```
terraform plan
```

You should see the new address outputted similar to:

```
Anycast IPv6 Address 2 = 2604:1380:2:2303::2
```

Keep in mind that the new address hasn’t be bound to the hosts.

Next Steps

Once you’ve created the new address, proceed to Lab 8

./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 08 - Adding an additional Anycast

Goal

>> Master the Terraform and auto configure the hosts with the new Anycast address

Prerequisites

>> Access to the lab master server from Lab 01

Lab Master Access

With your assigned lab username (i.e. bgp03), log into the lab master server using the your assigned lab and the password. You'll need to use a SSH client (i.e. PuTTy).

```
ssh <your_lab_username>@<lab_master_server>
```

Create the new Anycast address

Edit `ipv6.tf` and create a new variable called `anycast_addr_2` that uses the ::2 address in the IPv6 subnet.

```
anycast_addr_1      = "${cidrhost(local.anycast_network,1)}"
anycast_addr_2      = "${cidrhost(local.anycast_network,2)}"
```

Configure the new Anycast on the hosts

We'll need a new template that will apply the new address to the host. This new file would be "templates/enable_anycast_2.tpl". You can copy "templates/enable_anycast_1.tpl" as a base and edit.

```
cp templates/enable_anycast_1.tpl templates/enable_anycast_2.tpl
vi templates/enable_anycast_2.tpl
```

It should look like:

```
auto lo:1
iface lo:1 inet6 static
   address ${anycast_ip_2}
   netmask 64
EOF

ifup lo:1
service bird6 restart
```

Define the template variables

We'll need new Terraform files that will define the variables and execute the template on the host. Simple use the existing files for the first anycast address and modify the variables within to use the ::2 address.

```
cp enable_anycast_1.tf enable_anycast_2.tf
vi enable_anycast_2.tf
```

It should look like:

```
data "template_file" "enable_anycast_2" {
    template = "${file("templates/enable_anycast_2.tpl")}"
    vars = {
      anycast_ip_2     = "${local.anycast_addr_2}"
    }
}

resource "null_resource" "enable_anycast_2" {

    depends_on = ["null_resource.bird"]

    count = "${var.instance_count}"

    triggers = {
        template = "${data.template_file.enable_anycast_2.rendered}"
    }

    connection {
```

```
        type = "ssh"
        host = "${element(packet_device.hosts.*.access_public_ipv4,count.index)}"
        private_key = "${file("mykey")}"
        agent = false
    }

    provisioner "file" {
        content = "${data.template_file.enable_anycast_2.rendered}"
        destination = "/tmp/enable_anycast_2.sh"
    }

    provisioner "remote-exec" {
        inline = [
            "chmod +x /tmp/enable_anycast_2.sh",
            "/tmp/enable_anycast_2.sh"
        ]
    }
}
```

Plan and Apply Terraform

Review and execute the Terraform.

```
terraform plan  | more
terraform apply --auto-approve
```

Verify the new Anycast address

The new Anycast address should respond to curl requests.

```
curl http://[Anycast IPv6 Address 2]/
```

Verify the IPv6 Networking

Log into the deployed host and examine the networking.

```
terraform output
ssh root@<Server IP 4> -i mykey
ip -6 a show
```

Check that the new IPv6 Anycast address (::2) is assigned.

Next Steps

Once you're confident that the new address is working, proceed to Lab 9

./ BGP-Anycast-Workshop

Workshop teaching basics of BGP HA, Terraform automation, and IPv6.

View on GitHub

Lab 09 - Lab Cleanup

Prerequisites

>> Access to the lab master server from Lab 01

Lab Teardown

Congrats! You made it through the lab exercises!

Please take a moment to tear down (turn off) all the deployed hosts so they can be made available to others.

```
terraform destroy
```

Please mark your master lab account on the etherpad as complete so it can be deallocated.

Feedback

Do you have questions or feedback about the lab? Please leave them as an "Issue" on GitHub:

Workshop GitHub