Routing in an
Internet Service Provider

or - *How internet really works.*

Who we are.

Federico Favaro

Solution Architect, daily dealing with virtualization, routing and network automation with Python and Ansible. Software development background (in love with Elixir/Erlang).

https://www.linkedin.com/in/fedefava/

Stefano Sasso

Network Architect with focus on SDN, routing and switching, network automation and modern data center networking. Worked for an ISP. Pro-bono admin of AS 42463.

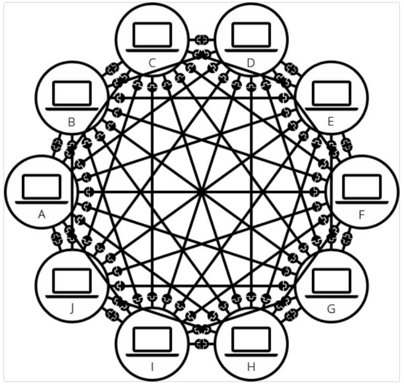
https://www.linkedin.com/in/ssasso/

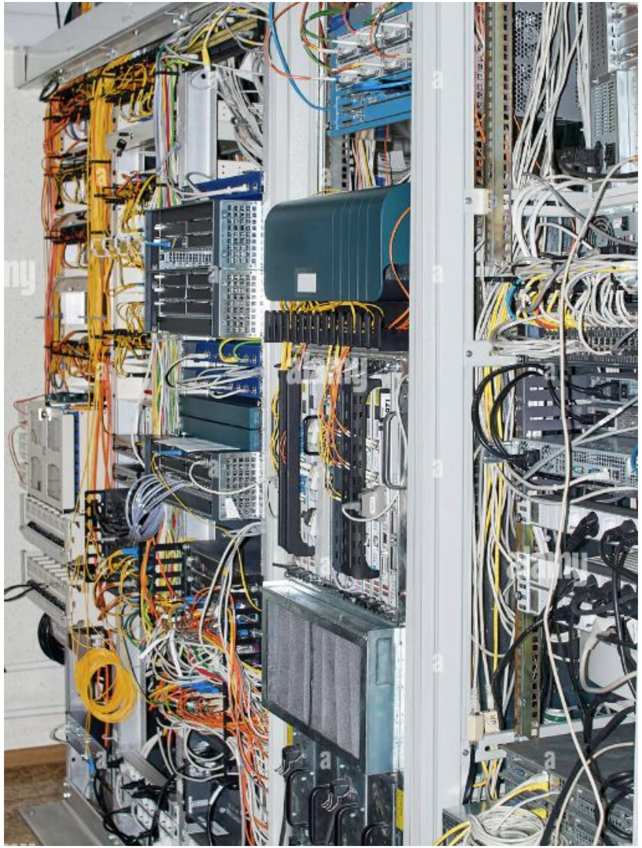
Internet behind the scenes....

- Ever wondered how you reach Facebook, YouTube, Netflix from your home?
- Where do your packets go “after” your home modem/router?

We’re here to (kind of*) answer you!

(*) See next slide





Spoiler alert.

Internet, and routing for any Internet Service Provider, is a very complex topic.

It would be impossible to give an overview about all the involved technologies: during this workshop a lot of stuff will be presented as oversimplified.

We will mainly focus on:

- ISP Internal (simple) backbone, based on OSPF.
- BGP and interconnection with the external world.

Some prerequisites...

- Some remembrance about IP Routing, OSPF and BGP
- Some remembrance about how *FRR* works (routing daemon used also in *Katharà*)
- A (free) github account - https://github.com/
- Willing to learn new stuff and to play with networking

Optionally:

- github cli installed on your computer, together with a SSH client (i.e., https://mobaxterm.mobatek.net/download.html)
- latest version of wireshark installed on your computer

let's start with a step back.

OSPF

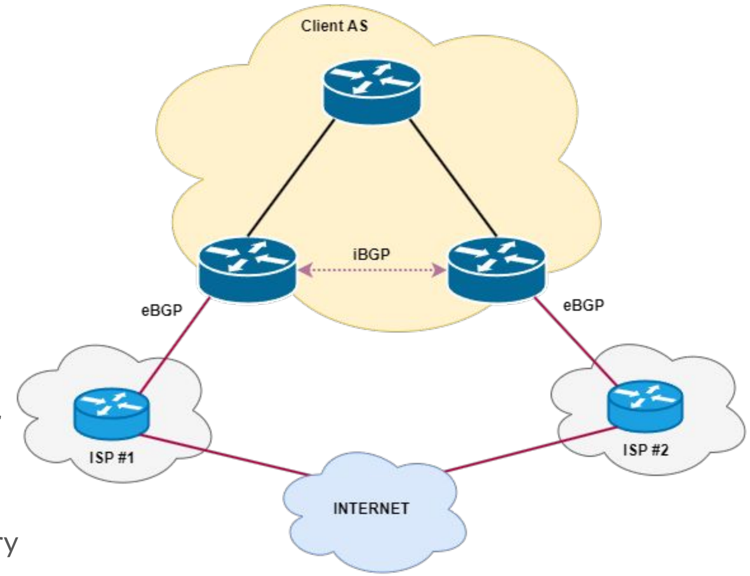
- IGP - Interior Gateway Protocol
- **Link-state** protocol based on
  - flooding to discover the network topology
  - Dijkstra algorithm to find the least cost path to reach a destination
- Network topology is “divided” in areas to increase scalability
- Mainly used to redistribute <u>**ONLY**</u> **internal** networks and **loopback addresses**
- ***Design goal is to minimise number of prefixes in IGP for scalability and rapid convergence***

LOOPBACK ADDRESSES ???

IP Addresses assigned to “virtual” (loopback) interfaces which are, by definition, ALWAYS UP

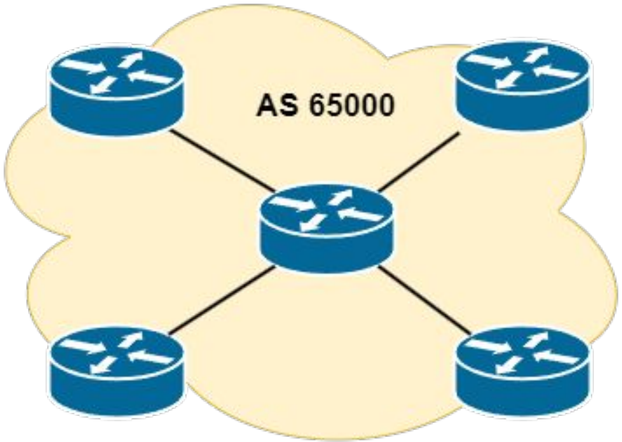
BGP

- EGP - Exterior Gateway Protocol
- Used for routing with other AS (Autonomous Systems)
- A lot of use cases... (MP-BGP anyone?)
  - For our scope, used to **redistribute operator networks to other operators,** and achieve redundancy and scalability.
  - Used to influence inbound and outbound routing with proper “filters” and “route maps” (BGP ATTRIBUTES associated to every route)
- Path vector protocol and incremental updates...
  - very scalable!
- Learns multiple paths via other BGP speakers, select the best paths and installs in the forwarding table



BGP - AS (Autonomous System)

- Network identified by a “unique number” (ASn)
- Single ownership, trust, administrative control and “routing policy”



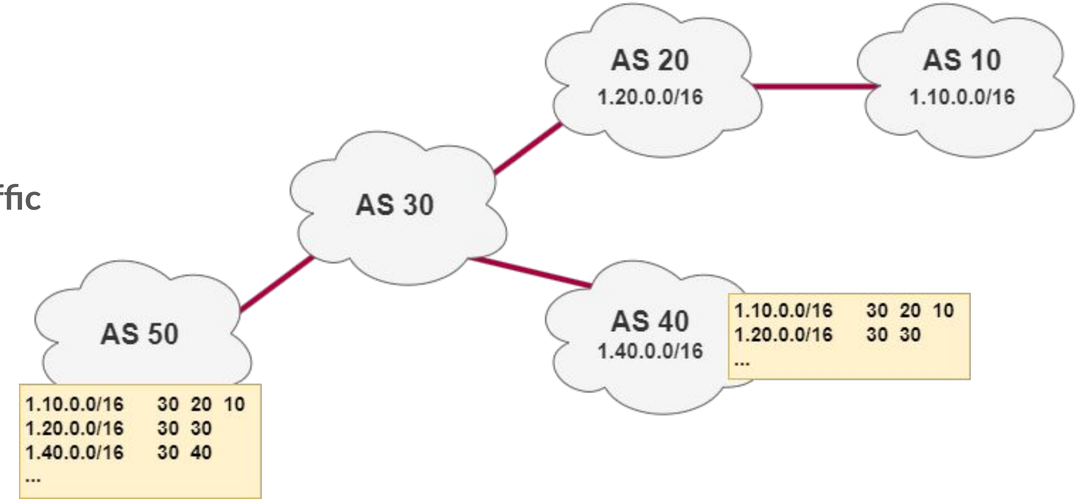
Common BGP Attributes

- **AS-Path**: It records the numbers of all ASs through which a route passes from the local end to the destination in the vector order.
- **Next-Hop**: IP Address of the next hop router to be used to route a packet.
- **Multi-Exit-Discriminator (MED)**: is an optional, non transitive, attribute (transmitted only between two neighboring ASs - the AS that receives the MED attribute will not advertise it to other ASs). Attribute used to influence the inbound traffic routing. ***Path with lowest MED wins.***
- **Local Preference**: Attribute that indicates the preference of a BGP route on a router. This attribute is valid only within the AS (not advertised to other ASs). Attribute used to influence the outbound traffic routing. ***Path with highest preference wins.***

AS Path

- **AS-Path**: It records the numbers of all ASs through which a route passes from the local end to the destination in the vector order.
- ***Path with shortest length wins.***

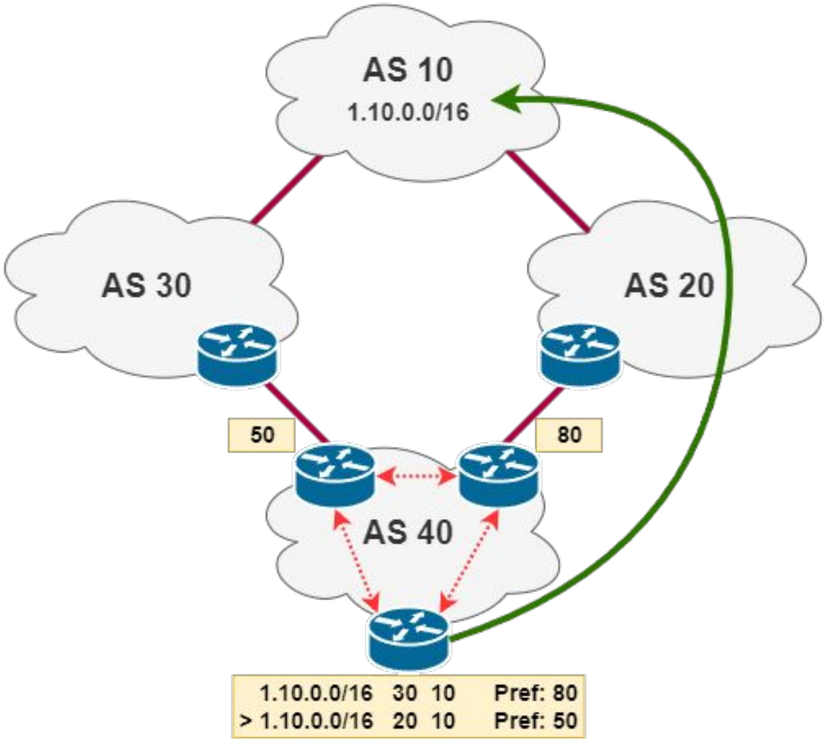
- **It is used to influence inbound traffic (applied to outbound announces)**



Local Preference

- **Local Preference**: Attribute that indicates the preference of a BGP route on a router. This attribute is valid only within the AS (not advertised to other ASs). Attribute used to influence the outbound traffic routing.
- ***Path with highest preference wins.***

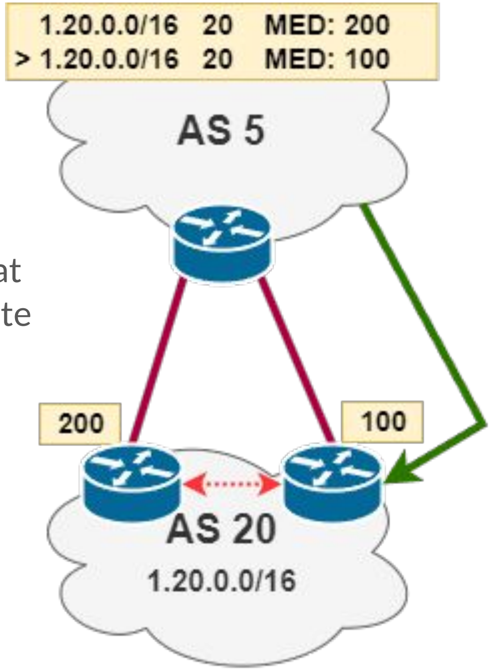
- **It is used to influence outbound traffic (applied to inbound announces)**



MED

- **Multi-Exit-Discriminator (MED)**: is an optional, non transitive, attribute (transmitted only between two neighboring ASs - the AS that receives the MED attribute will not advertise it to other ASs). Attribute used to influence the inbound traffic routing.
- ***Path with lowest MED wins.***

- **It is used to influence inbound traffic (applied to outbound announces)**



Multiple Routes for the same destination?

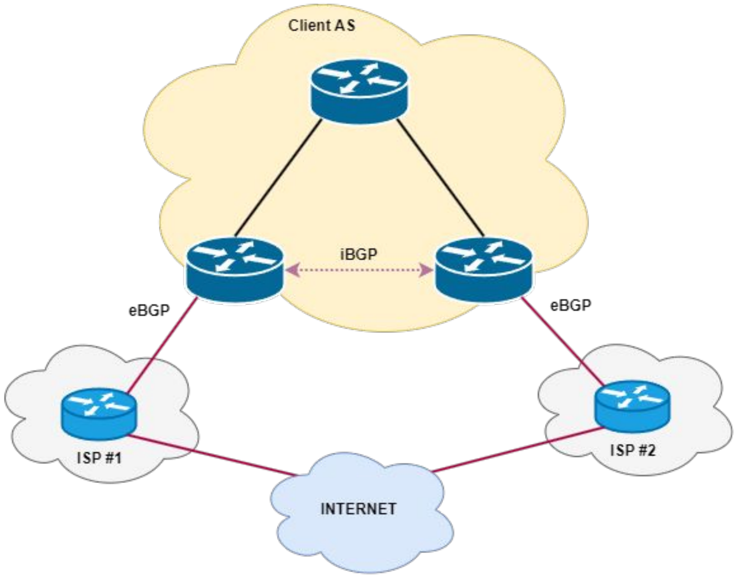
- Routing is based on LONGEST PREFIX MATCH
- However, **multiple routes with the same prefix** can be present on a router routing table.
- How the router selects the **best route for a destination**?

BGP Best Path Selection

1. **Weight** (local attribute, cisco proprietary):
   Prefer highest weight.
2. **Local Preference** (within AS):
   Prefer highest local preference.
3. **Originate** (local attribute):
   Prefer routes "created" by the BGP process itself over routes redistributed by/from other protocols (*next-hop 0.0.0.0*).
4. **AS path length**:
   Prefer the shortest path length.
5. **Origin** (legacy attribute).
6. **MED** (between AS):
   Prefer the lowest MED.
7. **eBGP path over iBGP path** (protocol metric):
   Prefer eBGP routes over iBGP.
8. **Shortest IGP path to BGP next hop**:
   Prefer the lowest IGP metric.
9. **Oldest path**:
   Prefer the path that was received first.
10. **Router ID**:
    Prefer the path received by the BGP neighbor with lowest Router ID.
11. **Neighbor IP address**:
    Prefer the path received by the BGP neighbor with lowest IP address.
    (this will only happen if you have two links between same routers)

eBGP vs iBGP

- Same protocol, different name and behaviour
- eBGP: peering between different AS
  - exchange prefixes with other ASes
  - “implement routing policy”
- iBGP: peering within the same AS
  - transport internet prefixes across backbone
  - can be used for local/customer prefixes

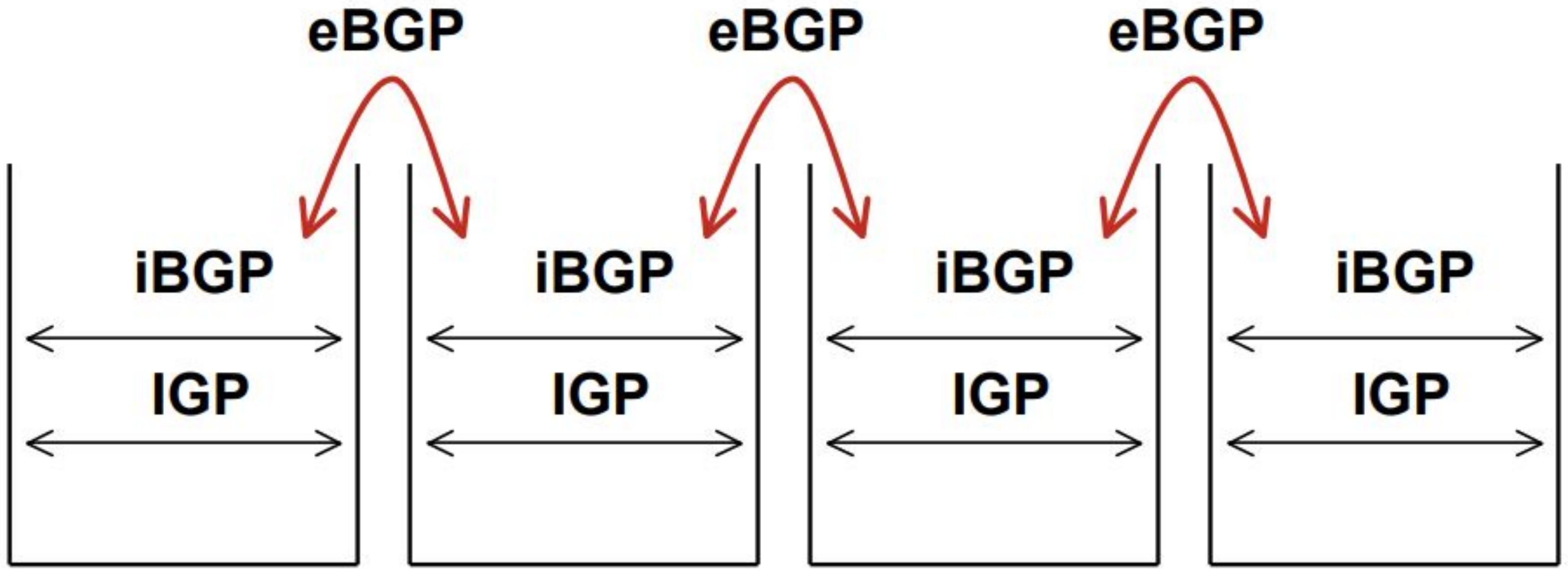


Why do we need iBGP?

BGP is more scalable than OSPF, and the internet routing table, which needs to be known by “internet speaking routers”, is very huge.

All the routers in a packet transit path must know how to route it.

eBGP vs iBGP



eBGP vs iBGP - need to know:

eBGP:

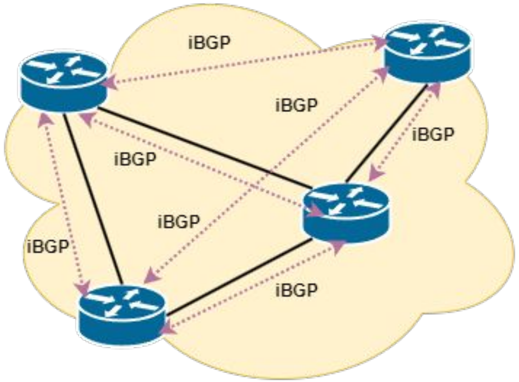
- when announcing a route, the next hop is set to local IP used for peering
- by default, BGP signaling packets have TTL set to 1
- any route received by an eBGP peer will be announced to any other eBGP and iBGP peer

iBGP:

- when announcing a route, the next hop is left unchanged
- by default, BGP signaling packets have TTL >> 1 *(vendor dependant)*
- any route received by an iBGP is NOT announced to other iBGP peers (but only to eBGP)

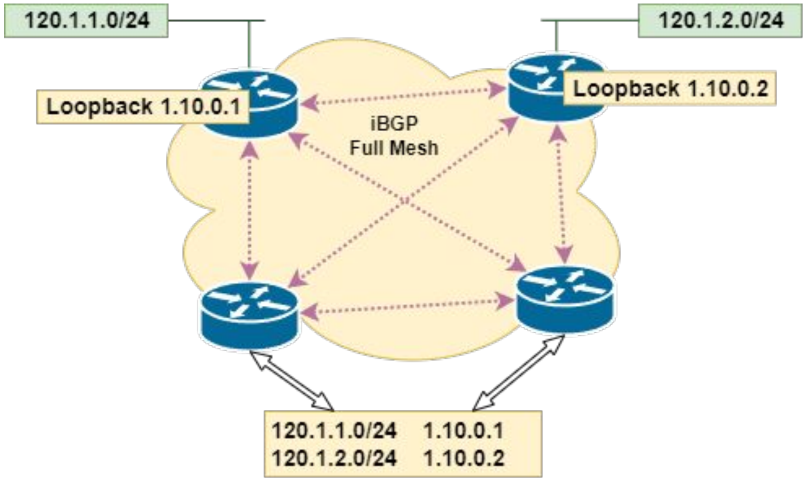
iBGP - Internal BGP

- BGP peering within the same AS, to internally exchange the routes learned by other eBGP peers
- Not required to be directly connected: IGP takes care of inter-BGP speaker connectivity
  - iBGP sessions between loopback interfaces! they will never go down, and are announced using i.e., OSPF - and we do not want BGP session to depend on state of a single interface or the physical topology
- iBGP routers must be **fully meshed**:
  - They pass on prefixes learned by other eBGP speakers
  - They do not pass on prefixes learned by other iBGP speakers
  - Each iBGP router must peer with every other iBGP speaker in the AS (topology independent)



iBGP - Internal BGP

The next-hop for a route announced via iBGP can be set to the router loopback address (recursive route lookup) → we do not need to know the “external” topology



Scaling iBGP

iBGP routers "*must*" be **fully meshed**…

Number of sessions for each iBGP speaker: $\frac{1}{2}n(n-1)$

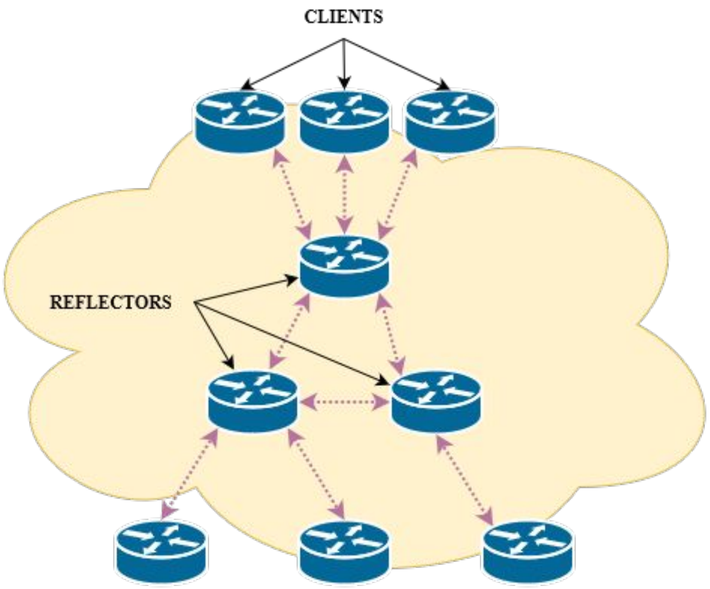
10 routers → 45 iBGP sessions… 100 routers → 4950 sessions.

Every router must have a session with the other 99 routers. And handle their messages and updates, and flaps.

Route Reflector

1. Reflector receives paths from clients and non-clients
2. Selects best path
3. If best path is from client, reflect to other clients and non-clients
4. If best path is from non-client, reflect to clients only

NON-MESHED clients. iBGP full-mesh only between route reflectors.



Some other terminology

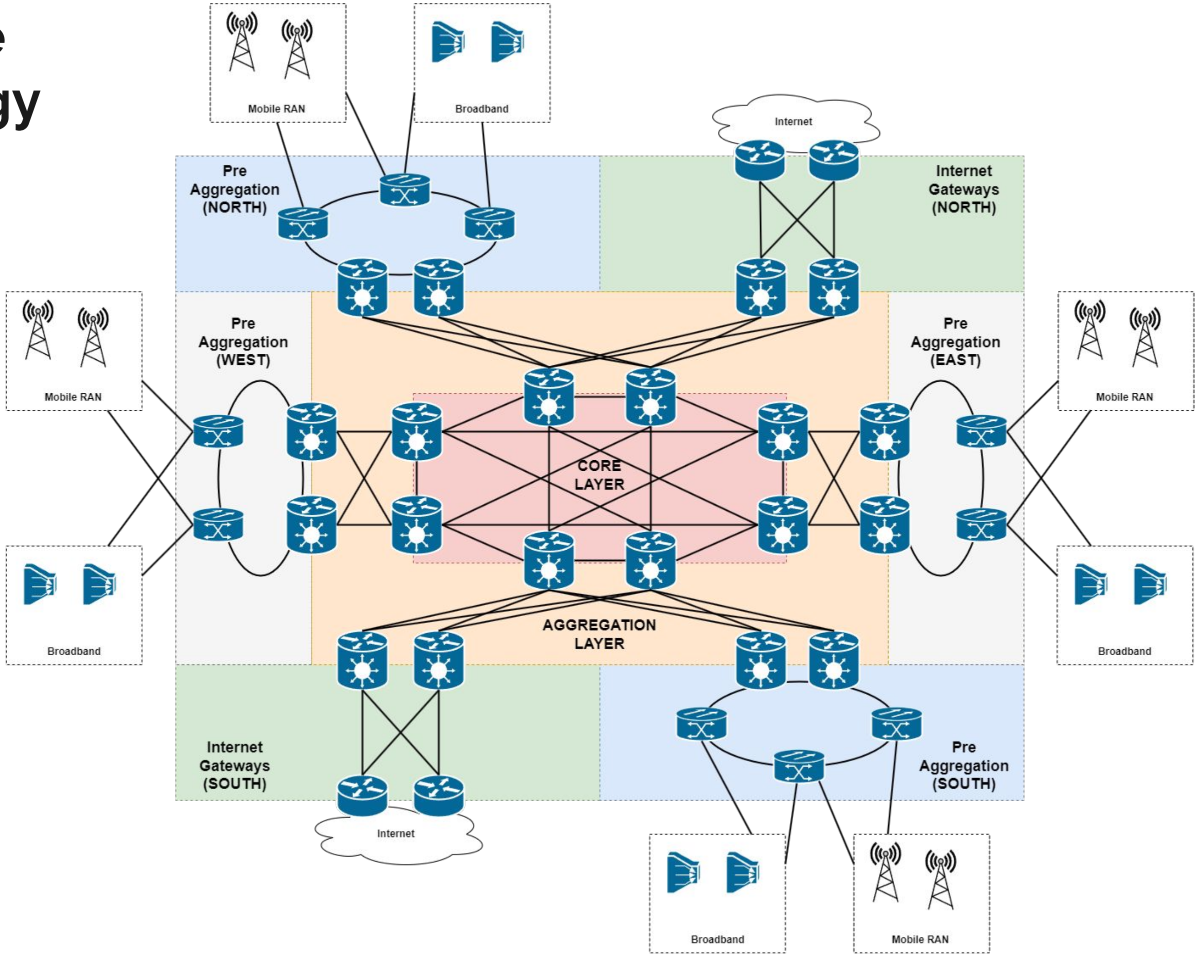
- **BGP Peering**: BGP session between two BGP speakers

"Types" of BGP Peering:

- **Upstream/Transit**: other BGP provider/AS which provides a "default route" or a full internet routing table (*usually paid*).
- **Peer**: other BGP provider/AS which provides routes only for its own networks (*usually free*).

- **IXP/Internet Exchange**: location where multiple providers create peerings between each other.
    - Some examples in Italy: NAMEX (Roma) - https://www.namex.it/, MIX-IT (Milano) - https://www.mix-it.net/, VSIX (Padova) - https://www.vs-ix.org/.

Sample topology



Welcome to...

ACME ISP

Your new Internet Service Provider

ACME

ACME is a new ISP in your city.

It has just obtained its own public resources (AS number, Public IP Addresses subnets) to start delivering services to its customers.

ACME wants to build its own backbone with state-of-the-art design, in a redundant and scalable manner

ACME has a contract with two Transit providers, and will be present at a local Internet Exchange for peering with other operators and content providers.

MEMO*: a lot of stuff will be presented as oversimplified.*

ACME

AS Number: 64666

IPv4 Public Subnet: 100.100.0.0/20

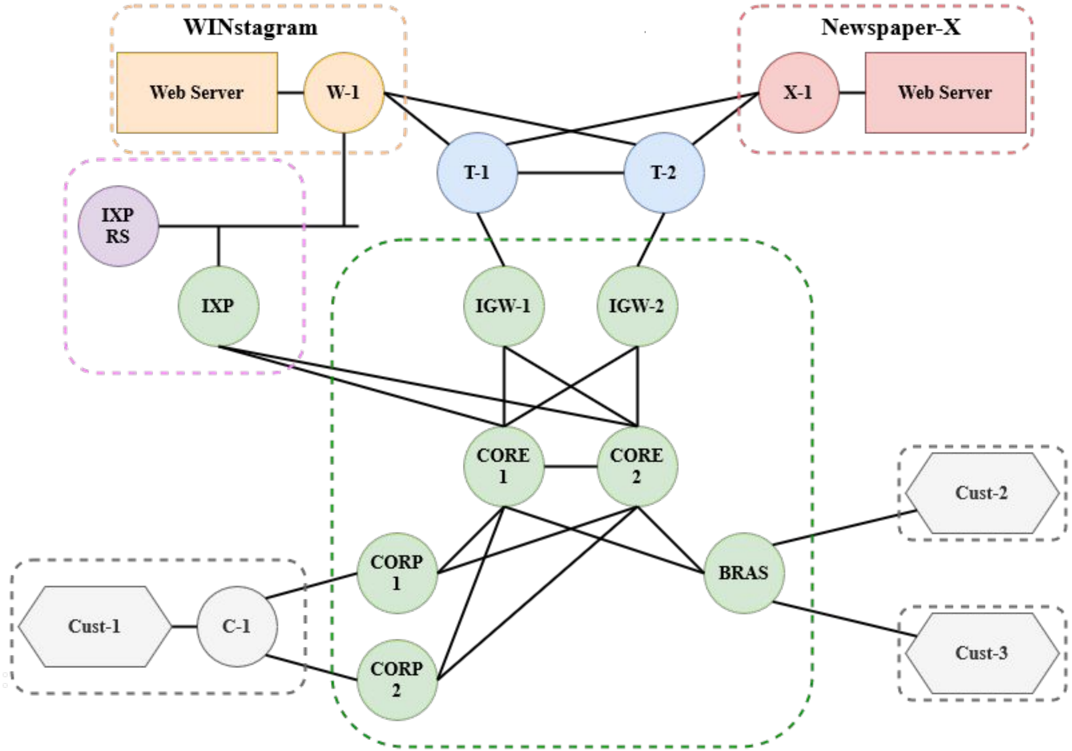
IPv6 Public Subnet: *let's keep it for the next time :-)*

TIP:
Using public IPv4 prefixes in a lab is bad form, so we'll pretend the *Shared Address Space prefix - RFC6598* (100.64.0.0/10) belongs to public address space. Same for AS numbers: we will use the range *from 64512 to 65535*.

Other Actors

- **Transit-1 and Transit-2**
  - Transit/Upstream operators for ACME network. Transit-1 costs less than Transit-2, so it needs to be preferred for the traffic path.
- **WINstagram: *The new winning social network.***
  - This is a new social. ACME expects lot of traffic to/from this AS, so it will create a "peering relationship" at the local Internet Exchange to save on the traffic costs.
- **Newspaper-X**
  - Countrywide newspaper.
- **(Big) Customer 1**
  - Business customer. Requires Highly available internet active/backup connection from ACME (fiber with microwave as backup), with static IP addresses.
- **Customer 2, Customer 3**
  - Standard - Consumer customers with dynamic IP addresses *[simulated]*.

ACME High Level Design



ACME Routing policies (1)

Traffic exchanged at the IXP must be prioritized.

That will be easier: at an IXP, the direct peering allows to have a shortest AS path than any other transit provider.

NOTE: Some IXPs usually deploy a *Route Server* on the *Peering LAN*.

A Route Server is “similar to a route reflector for eBGP”, allowing to exchange routes without “full mesh peerings”. A Route Server will redistribute routes without changing the next hop and AS Path (off path).

NOTE: The same kind of routing policies we will apply to ACME can also apply to other “content providers” such as *WINstagram* and *Newspaper-X*.

ACME Routing policies (2)

Transit-1 costs less than Transit-2, so it needs to be preferred for the traffic path.

We can achieve that with:

- **Local Preference** for T1 greater than T2
- **AS-Path** for T1 shorter than T2
  - → AS-Path prepending to T2 (forcefully set a longer AS-Path)

NOTE: The same kind of routing policies we will apply to ACME can also apply to other “content providers” such as *WINstagram* and *Newspaper-X*.

ACME Routing policies (summary)

	Local Preference	AS-Path
Transit-1	100	*unchanged*
Transit-2	< *100* 50	*PREPEND 1 time*

NOTE: We assume that LocPref of 100 are the “defaults” for our network (assigned to Transit-1).

ACME Routing policies (BEST PRACTICES)

With no additional configuration, BGP routers propagate every route known to them to all eBGP neighbors, which means that your network propagates routes between Transit-1 and Transit-2 (and IXP).

You must filter BGP prefixes sent to Transit-1 and Transit-2 and advertise only prefixes with an **empty AS path – the prefixes originating in your autonomous system** (unless you offer transit services).

At the same time, it is a best practice to filter the prefixes you announce (or you receive from your customers), to avoid announcing (and receiving) bad prefixes. Only the aggregate of your network ***should*** be announced.
Additionally, you should remove any Private AS number used in your network (*** not for our lab activity*).

NOTE: The same kind of routing policies we will apply to ACME can also apply to other “content providers” such as *WINstagram* and *Newspaper-X*.

ACME Routing policies - (Big) Customer 1

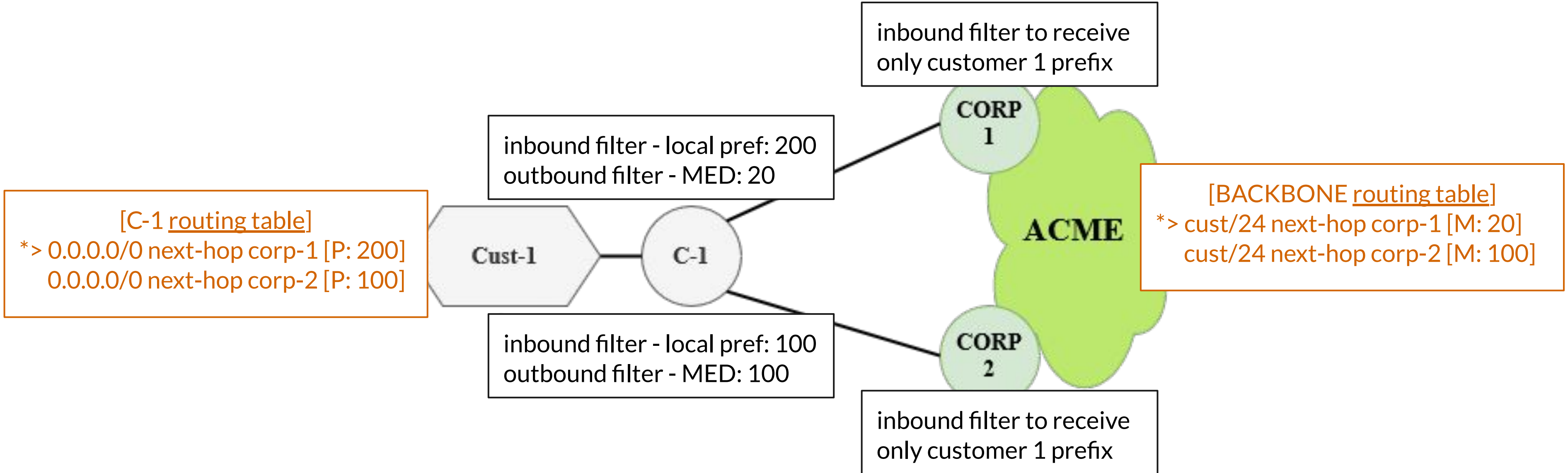
Requires Highly available internet active/backup connection from ACME (fiber with microwave as backup), with static IP addresses.

We can achieve that on the Customer-1 Router with:

- **Local Preference** for link to CORP-1 greater than CORP-2 (i.e., 200/100)
- **MED** for link to CORP-1 lower than CORP-2 (i.e., 20/100)

ACME Routing policies (in practice)

Customer-1 point of view

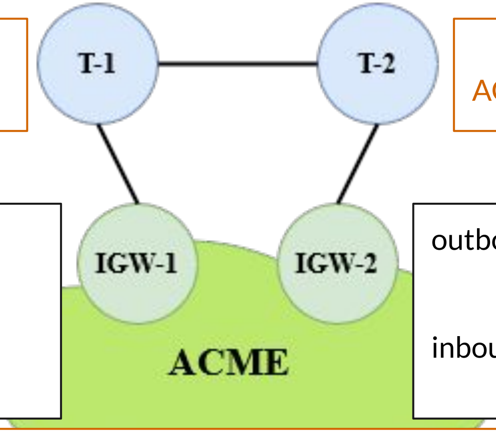


ACME Routing policies (in practice)

ACME point of view

[other router routing table]
*> ACME next-hop T-1 [AS-Path 65001 64666]
ACME next-hop T-2 [AS-Path 65002 64666 64666]

[T-1 routing table]
ACME next-hop igw-1 [AS-Path 64666]



[T-2 routing table]
ACME next-hop igw-2 [AS-Path 64666 64666]

outbound filter:
announce only aggregate

inbound filter:
set local preference: 100

outbound filter:
announce only aggregate
set as-path prepend * 1
inbound filter:
set local preference: 50

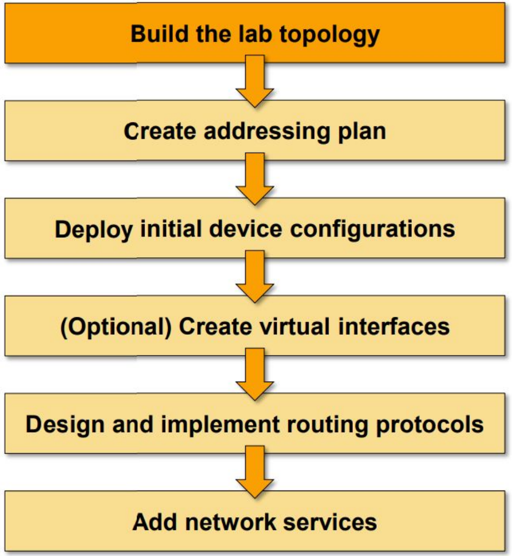
[BACKBONE routing table]
*> 0.0.0.0/0 next-hop transit-1 [P: 100]
0.0.0.0/0 next-hop transit-2 [P: 50]

LAB TIME!

And now we could...

- Create virtual routers/devices
- Manually configure every single link between them
- Manually configure IP addresses on every interface, and check that we configured the correct one

... and we will lose at least 1 hour only for these (repetitive) tasks.



Images are courtesy of Ivan Pepelnjak - ipSpace.net

Meet netlab.

We can:

- Create a high level description of the network as a YAML file
- Save the file
- Execute a command and wait for the virtual devices to appear.
- (Have the devices configured for us with OSPF, BGP, …)

Using *infrastructure-as-a-code* methodology.

topology.yml

```yaml
nodes: [ r1, r2 ]

defaults.device: eos
provider: clab

module: [ ospf ]

links: [ r1, r2, r1-r2, r1-r2 ]
```

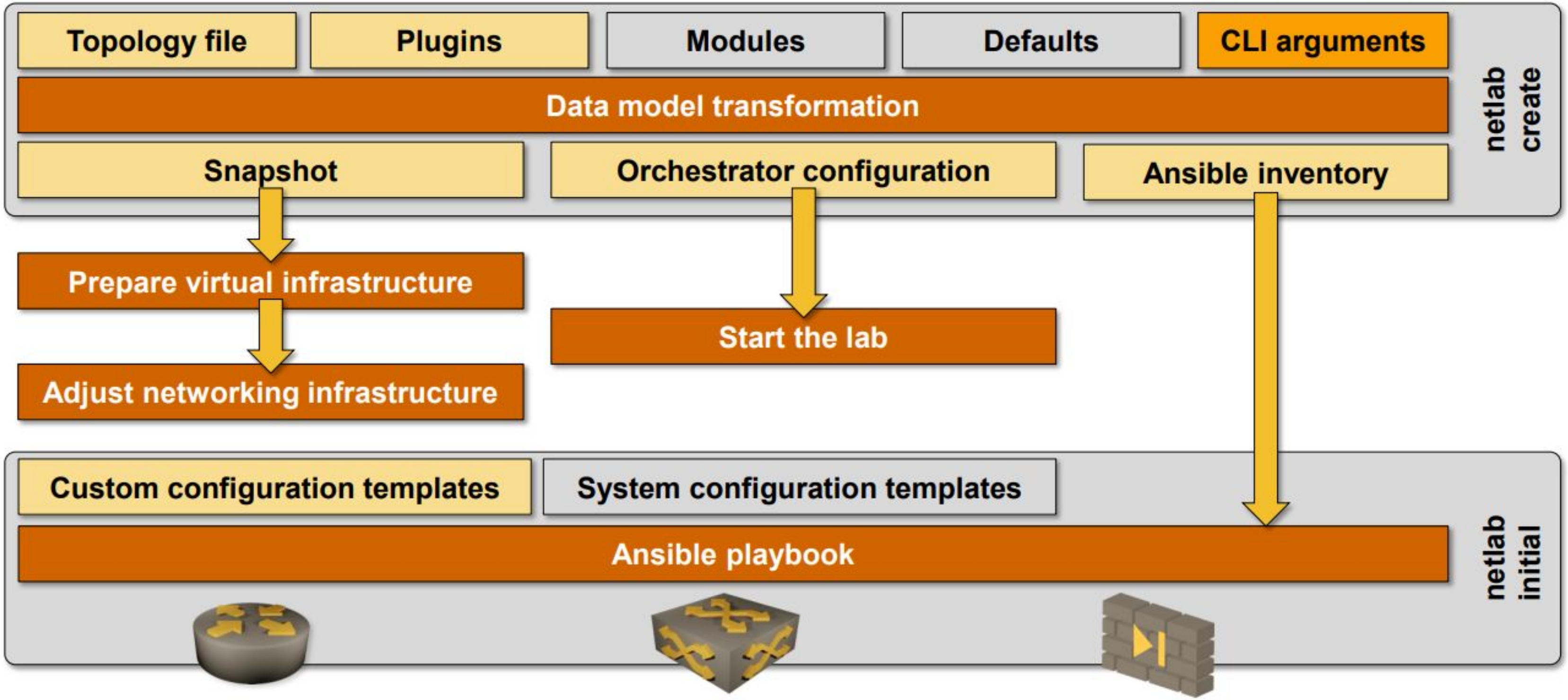
Images are courtesy of Ivan Pepelnjak - ipSpace.net

Meet netlab.

https://netlab.tools/

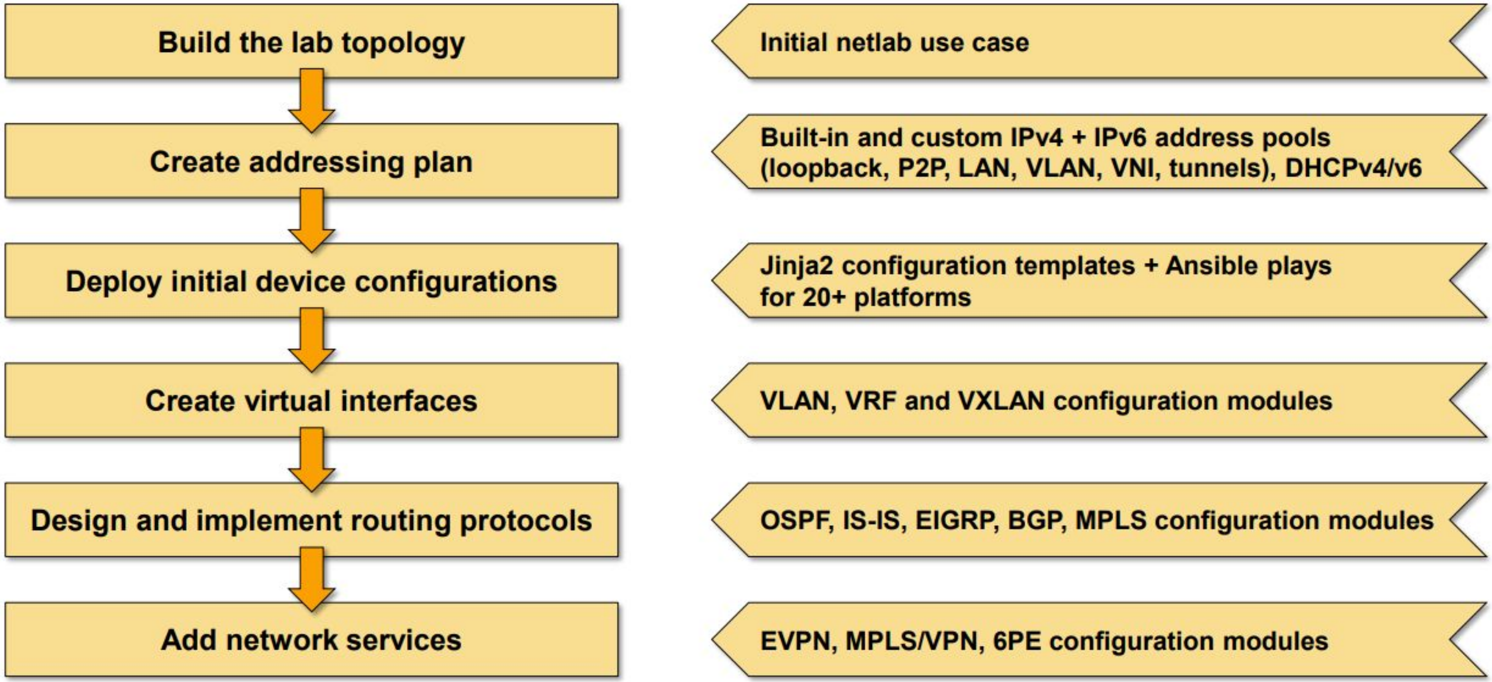
netlab will help you be more proficient once you decide to drop GUI-based virtual networking labs and build your labs using CLI and infrastructure-as-code principles.

netlab behind the scenes



Images are courtesy of Ivan Pepelnjak - ipSpace.net

netlab behind the scenes



Images are courtesy of Ivan Pepelnjak - ipSpace.net

ACME ISP with netlab

- For this workshop, we can create ACME ISP topology using netlab
- We will use only the basic functions of netlab:
    - Creation of **FRR** **(*)** and other Linux containers
    - Creation of interfaces and links
    - IP Address allocation to the different interfaces

- **OSPF and BGP configuration is part of this workshop!**

(* you should already be familiar with it: it's the same daemon used in *Katharà*)
(and yes, at the end you can have the full configs as reference :-))

ACME ISP with netlab

We can start the netlab topology using GitHub Codespaces.

NOTE: you will need a github account for this.

⇒ Let’s see how to start the Lab environment.

(optional: collaborative working)

If you want, you can create only one codespace for your team, and grant access (**via SSH**) to other team members.

→ different people can take care of different configurations.

Prerequisite: github cli and ssh client on team members' computer.

Start a codespace for the lab activities:

Open this URL:

https://bit.ly/acme-isp

(https://github.com/codespaces/new/ssasso/acme-isp)

→ Click on "**Create codespace**"

Start a codespace for the lab activities:

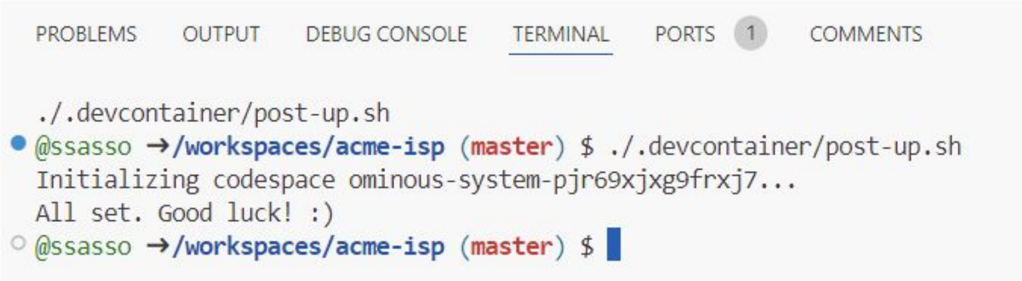
A new window, looking similar to VSCode, will open.

(it can take some minutes to complete the initialization process)

When the shell on the bottom part will say…

```
Initializing codespace xxxxxxxx…
All set. Good luck! :)
```

… the environment is ready for playing.



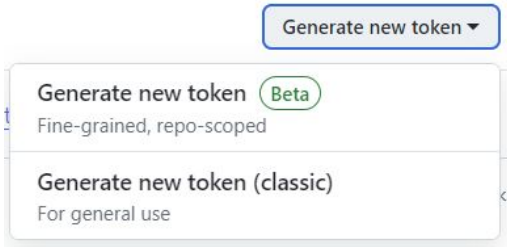
(collaborative working: how to do it) [1]

You can enable your team to access you account and codespace using a temporary access token.

From the github interface:

→ Profile Settings → Developer Settings → Personal access tokens (classic)

→ Generate new token (classic)



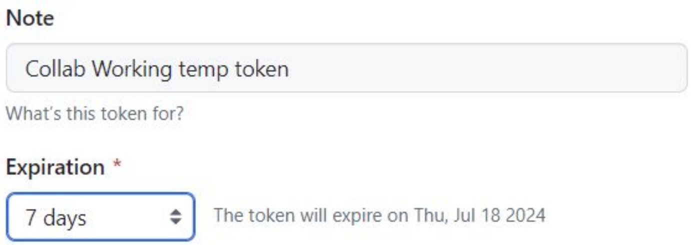
https://github.com/settings/tokens/new

(collaborative working: how to do it) [2]

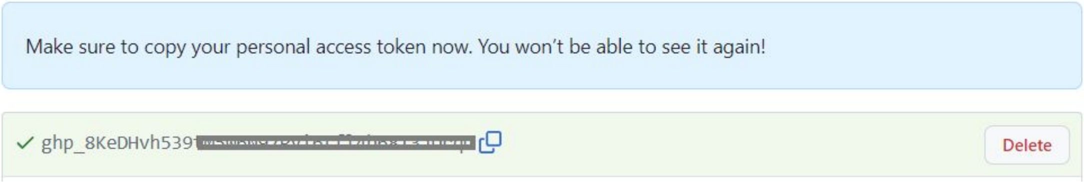
Set a token name, and a short expiration time (i.e., 7 days)

Select at least:

- repo (all settings)
- read:org
- codespace (all settings)



Copy (and share) the generated token.



(collaborative working: how to do it) [3]

On the computer with the github cli installed:

```
$ gh auth login -s codespace
```

(insert the required data and token string)

```
$ gh codespace ssh
```

(select the right codespace)

(collaborative working: how to do it) [4]

```
[root@rockystone ~]# gh auth login -s codespace
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? No
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org'.
? Paste your authentication token: ****************************************
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as ssasso
```

(collaborative working: how to do it) [5]

```
[root@rockystone ~]# gh codespace ssh
? Choose codespace: ssasso/acme-isp (master): super palm-tree
Linux codespaces-c6d4cc 6.5.0-1022-azure #23~22.04.1-Ubuntu SMP Thu May  9 17:59:24 UTC 2024 x86_64

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.
@ssasso ➜ /workspaces/acme-isp (master) $
```

REMEMBER TO DELETE THE CODESPACE AFTER THE LAB ACTIVITIES!

Manage your codespaces at:
https://github.com/codespaces

netlab quickstart

- cd into the topology directory
- execute

  `netlab up`
- you can connect to any virtual device shell with

  `netlab connect xxx`
- check the status of the virtual devices defined in the topology

  `netlab status`
- execute lab validation rules (if someone has written them for the lab topology ;))

  `netlab validate`
- to shut down everything you can use

  `netlab down`

https://netlab.tools/netlab/cli/

```
@ssasso →/workspaces/acme-isp/T3-full-cfg (master) $ netlab status
Lab default in /workspaces/acme-isp/T3-full-cfg
  status: started
  provider(s): clab
```

node	device	image	mgmt IPv4	connection	provider	VM/container	status
bras	frr	quay.io/frrouting/frr:9.1.0	192.168.121.116	docker	clab	clab-T3-full-cfg-bras	Up 2 minutes
c1host	linux	ssasso/netlab-linux-host:latest	192.168.121.121	docker	clab	clab-T3-full-cfg-c1host	Up 2 minutes
c1rt	frr	quay.io/frrouting/frr:9.1.0	192.168.121.120	docker	clab	clab-T3-full-cfg-c1rt	Up 2 minutes
c2host	linux	ssasso/netlab-linux-host:latest	192.168.121.122	docker	clab	clab-T3-full-cfg-c2host	Up 2 minutes
c3host	linux	ssasso/netlab-linux-host:latest	192.168.121.123	docker	clab	clab-T3-full-cfg-c3host	Up 2 minutes
core1	frr	quay.io/frrouting/frr:9.1.0	192.168.121.101	docker	clab	clab-T3-full-cfg-core1	Up 2 minutes
core2	frr	quay.io/frrouting/frr:9.1.0	192.168.121.102	docker	clab	clab-T3-full-cfg-core2	Up 2 minutes
corp1	frr	quay.io/frrouting/frr:9.1.0	192.168.121.114	docker	clab	clab-T3-full-cfg-corp1	Up 2 minutes
corp2	frr	quay.io/frrouting/frr:9.1.0	192.168.121.115	docker	clab	clab-T3-full-cfg-corp2	Up 2 minutes
igw1	frr	quay.io/frrouting/frr:9.1.0	192.168.121.111	docker	clab	clab-T3-full-cfg-igw1	Up 2 minutes
igw2	frr	quay.io/frrouting/frr:9.1.0	192.168.121.112	docker	clab	clab-T3-full-cfg-igw2	Up 2 minutes
ixp	frr	quay.io/frrouting/frr:9.1.0	192.168.121.113	docker	clab	clab-T3-full-cfg-ixp	Up 2 minutes
nxrt	frr	quay.io/frrouting/frr:9.1.0	192.168.121.150	docker	clab	clab-T3-full-cfg-nxrt	Up 2 minutes
nxweb	linux	ssasso/pyweb:latest	192.168.121.155	docker	clab	clab-T3-full-cfg-nxweb	Up 2 minutes
transit1	frr	quay.io/frrouting/frr:9.1.0	192.168.121.131	docker	clab	clab-T3-full-cfg-transit1	Up 2 minutes
transit2	frr	quay.io/frrouting/frr:9.1.0	192.168.121.132	docker	clab	clab-T3-full-cfg-transit2	Up 2 minutes
winsrt	frr	quay.io/frrouting/frr:9.1.0	192.168.121.140	docker	clab	clab-T3-full-cfg-winsrt	Up 2 minutes
winsweb	linux	ssasso/pyweb:latest	192.168.121.144	docker	clab	clab-T3-full-cfg-winsweb	Up 2 minutes

https://netlab.tools/netlab/cli/

FRR quickstart

After `netlab connect`, you will enter the FRR virtual device linux shell.

You can enter the FRR shell just typing:

`vtysh`

```
@ssasso →/workspaces/acme-isp/T3-full-cfg (master) $ netlab connect ixp
Connecting to container clab-T3-full-cfg-ixp, starting bash
Use vtysh to connect to FRR daemon

ixp(bash)#vtysh

Hello, this is FRRouting (version 9.1_git).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

ixp# show ip bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 100.100.0.13, local AS number 64666 vrf-id 0
BGP table version 99
RIB entries 33, using 3168 bytes of memory
Peers 3, using 39 KiB of memory

Neighbor              V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
winsrt(100.70.70.2)   4      65040       105       115       99    0    0 00:02:31            1        1 winsrt
core1(100.100.0.1)    4      64666       121        77       99    0    0 00:02:31           25        2 core1
core2(100.100.0.2)    4      64666       121        77       99    0    0 00:02:31           25        2 core2

Total number of neighbors 3
ixp#
```

FRR quickstart

- enter configuration mode:
  `configure`
- show IP routing table:
  `show ip route`
- show BGP routes:
  `show ip bgp`
- show BGP neighbors summary
  `show ip bgp summary`

```
ixp# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

K>* 0.0.0.0/0 [0/0] via 192.168.121.1, eth0, 00:05:44
C>* 100.70.70.0/24 is directly connected, eth3, 00:05:18
B>  100.71.0.0/16 [200/0] via 100.100.0.11 (recursive), weight 1, 00:03:44
  *                         via 100.100.1.21, eth1, weight 1, 00:03:44
  *                         via 100.100.1.25, eth2, weight 1, 00:03:44
B>  100.71.1.1/32 [200/0] via 100.100.0.11 (recursive), weight 1, 00:03:44
  *                         via 100.100.1.21, eth1, weight 1, 00:03:44
  *                         via 100.100.1.25, eth2, weight 1, 00:03:44
B>  100.72.0.0/16 [200/0] via 100.100.0.11 (recursive), weight 1, 00:03:44
  *                         via 100.100.1.21, eth1, weight 1, 00:03:44
  *                         via 100.100.1.25, eth2, weight 1, 00:03:44
B>  100.72.2.2/32 [200/0] via 100.100.0.11 (recursive), weight 1, 00:03:44
  *                         via 100.100.1.21, eth1, weight 1, 00:03:44
  *                         via 100.100.1.25, eth2, weight 1, 00:03:44
B>* 100.80.44.0/24 [20/20] via 100.70.70.2, eth3, weight 1, 00:03:44
B>  100.90.55.0/24 [200/0] via 100.100.0.11 (recursive), weight 1, 00:03:44
  *                          via 100.100.1.21, eth1, weight 1, 00:03:44
  *                          via 100.100.1.25, eth2, weight 1, 00:03:44
B>  100.100.0.0/20 [200/0] via 100.100.0.1 (recursive), weight 1, 00:03:44
  *                          via 100.100.1.21, eth1, weight 1, 00:03:44
                           via 100.100.0.2 (recursive), weight 1, 00:03:44
  *                          via 100.100.1.25, eth2, weight 1, 00:03:44
O>* 100.100.0.1/32 [110/10] via 100.100.1.21, eth1, weight 1, 00:04:17
O>* 100.100.0.2/32 [110/10] via 100.100.1.25, eth2, weight 1, 00:04:22
O>* 100.100.0.11/32 [110/20] via 100.100.1.21, eth1, weight 1, 00:04:17
  *                          via 100.100.1.25, eth2, weight 1, 00:04:17
O>* 100.100.0.12/32 [110/20] via 100.100.1.21, eth1, weight 1, 00:04:17
  *                          via 100.100.1.25, eth2, weight 1, 00:04:17
```

repository structure

- **T0-empty**
  This topology creates **only** the virtual devices and manages the addressing of the different interfaces, including the loopbacks. It can be used as a starting point if you want to configure everything else (i.e., OSPF, BGP) from scratch.

- **T1-core-only**
  This topology creates all the virtual devices, all the interfaces and related addressing. It configures OSPF and BGP on both Core Routers and *CORP-1/CORP-2*. It will allow you to configure BGP and related policies on Customer-1 router (login to core1 to check the routes, or use "`netlab validate`" **).

NOTE: you cannot launch multiple topologies at the same time. But you can launch a lab, collect all the generated configs with "`netlab collect`", and shut it down.

repository structure

- **T2-empty-acme**
  This topology creates all the virtual devices, all the interfaces and related addressing. It configures BGP on <u>all the devices except ACME's</u>.
  It will allow you to configure on your own OSPF, BGP and related policies on the ACME devices (login to any other device to check the routes, or use "`netlab validate`" **).

- **T3-full-cfg**
  This topology creates the full lab configuration: virtual devices, addressing, OSPF and BGP configuration. You can use this as a reference deployment to check the full configuration.

NOTE: you cannot launch multiple topologies at the same time. But you can launch a lab, collect all the generated configs with "`netlab collect`", and shut it down.

Let’s get our hands dirty

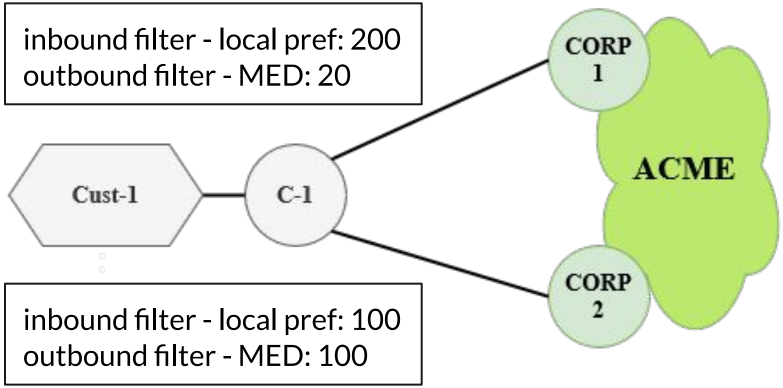
T1-core-only

- This topology creates all the virtual devices, all the interfaces and related addressing.
- It configures OSPF and BGP on both Core Routers and CORP-1/CORP-2.
- **It will allow you to configure BGP and related policies on Customer-1 router**
- (login to core1 to check the routes, or use “netlab validate” **)

```
cd T1-core-only
netlab up
netlab connect c1rt
```

T1-core-only - configure “C-1” (c1rt)

Customer AS: 65535 (private AS for customer)
Customer subnet: 100.100.15.0/24



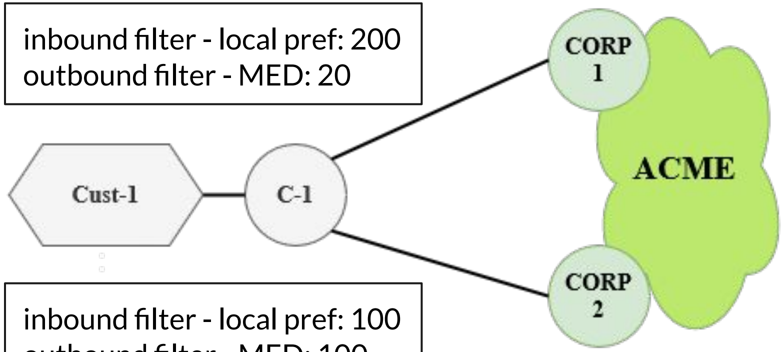

```
bgp as-path access-list 1 permit ^$
!
route-map corp1-out permit 10
 match as-path 1
 set metric 20
route-map corp1-out deny 99
!
route-map corp2-out permit 10
 match as-path 1
 set metric 100
route-map corp2-out deny 99
!
route-map corp1-in permit 10
 set local-preference 200
!
route-map corp2-in permit 10
 set local-preference 100
```

T1-core-only - configure "C-1" (c1rt)

Customer AS: 65535 (private AS for customer)
Customer subnet: 100.100.15.0/24

```
router bgp 65535
 neighbor 100.100.2.2 remote-as 64666
 neighbor 100.100.2.2 description corp1
 neighbor 100.100.2.6 remote-as 64666
 neighbor 100.100.2.6 description corp2
 !
 address-family ipv4 unicast
  network 100.100.15.0/24
  neighbor 100.100.2.2 activate
  neighbor 100.100.2.2 route-map corp1-in in
  neighbor 100.100.2.2 route-map corp1-out out
  neighbor 100.100.2.6 activate
  neighbor 100.100.2.6 route-map corp2-in in
  neighbor 100.100.2.6 route-map corp2-out out
 exit-address-family
```

inbound filter - local pref: 200
outbound filter - MED: 20



inbound filter - local pref: 100
outbound filter - MED: 100

T1-core-only - configure “C-1” (c1rt)

SCENARIO VALIDATION:

- show ip bgp <prefix> (on c1rt and core1/core2)
  - [c1rt] show ip bgp 100.100.0.0/20
  - [core1/2] show ip bgp 100.100.15.0/24
- netlab validate
- traceroute from *c1host*
- simple reachability check (ping, curl)

T1-core-only - configure “C-1” (c1rt) → validation

```
c1rt# show ip bgp 100.100.0.0/20
BGP routing table entry for 100.100.0.0/20
Paths: (2 available, best #2, table default)
  Not advertised to any peer
  64666
    100.100.2.6(corp2) from corp2(100.100.2.6) (100.100.0.15)
      Origin IGP, localpref 100, valid, external
      Last update: Sun Jul 14 12:55:44 2024
  64666
    100.100.2.2(corp1) from corp1(100.100.2.2) (100.100.0.14)
      Origin IGP, localpref 200, valid, external, bestpath-from-AS 64666,  best (Local Pref)
      Last update: Sun Jul 14 12:55:44 2024
```

number of paths, best one

AS Path received

next hop, received from (peer, router id)

received attributes

best path, reason

T1-core-only - configure “C-1” (c1rt) → validation

```
core1# show ip bgp 100.100.15.0/24
BGP routing table entry for 100.100.15.0/24
Paths: (2 available, best #2, table default)
  65535, (Received from a RR-client)
    100.100.0.15(corp2) (metric 10) from corp2(100.100.0.15) (100.100.0.15)
      Origin IGP, metric 100, localpref 10, valid, internal
      AddPath ID: RX 5, TX-All 31 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13)
corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 12:55:43 2024
  65535, (Received from a RR-client)
    100.100.0.14(corp1) (metric 10) from corp1(100.100.0.14) (100.100.0.14)
      Origin IGP, metric 20, localpref 10, valid, internal, bestpath-from-AS 65535,  best (MED)
      AddPath ID: RX 5, TX-All 32 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13)
corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 12:55:43 2024
```

number of paths, best one

AS Path received

next hop, received from (peer, router id)

received attributes

best path, reason

T1-core-only - configure “C-1” (c1rt) → validation

```
$ netlab validate
[... cut ...]

[c1_best_out] Check correct Best Path from Customer-1 Router [ node(s): c1rt ]
[PASS]        c1rt: The prefix 100.100.0.0/20 is in the BGP table with BGP router
ID=100.100.0.14,best path=True
[PASS]        Test succeeded

[c1_best_in]  Check correct Best Path to Customer-1 Router [ node(s): core1,core2 ]
[PASS]        core1: The prefix 100.100.15.0/24 is in the BGP table with BGP router
ID=100.100.0.14,best path=True,BGP MED=20
[PASS]        core2: The prefix 100.100.15.0/24 is in the BGP table with BGP router
ID=100.100.0.14,best path=True,BGP MED=20

[... cut ...]
```

T1-core-only - configure “C-1” (c1rt) → validation

```
c1host:/# traceroute nxweb

traceroute to nxweb (100.90.55.55), 30 hops max, 46 byte packets

 1   100.100.15.20 (100.100.15.20)  0.010 ms  0.008 ms  0.008 ms
 2   100.100.2.2 (100.100.2.2)  0.008 ms  0.009 ms  0.008 ms
 3   100.100.1.29 (100.100.1.29)  0.008 ms  0.028 ms  0.008 ms
 4   100.100.1.6 (100.100.1.6)  0.008 ms  0.009 ms  0.008 ms
 5   100.71.0.2 (100.71.0.2)  0.008 ms  0.008 ms  0.016 ms
 6   100.71.0.9 (100.71.0.9)  0.008 ms  0.008 ms  0.008 ms
 7   nxweb (100.90.55.55)  0.008 ms  0.008 ms  0.008 ms
```

c1rt

corp1

T1-core-only - configure “C-1” (c1rt) → validation

```
c1host:/# ping -c 2 nxweb

PING nxweb (100.90.55.55): 56 data bytes

64 bytes from 100.90.55.55: seq=0 ttl=58
time=0.183 ms
64 bytes from 100.90.55.55: seq=1 ttl=58
time=0.116 ms

--- nxweb ping statistics ---

2 packets transmitted, 2 packets received, 0%
packet loss
round-trip min/avg/max = 0.116/0.149/0.183 ms

c1host:/# curl nxweb

Welcome to Newspaper-X.
Your IP Address is: 100.100.15.21
```

```
c1host:/# ping -c 2 winsweb

PING winsweb (100.80.44.44): 56 data bytes

64 bytes from 100.80.44.44: seq=0 ttl=59
time=0.180 ms
64 bytes from 100.80.44.44: seq=1 ttl=59
time=0.127 ms

--- winsweb ping statistics ---

2 packets transmitted, 2 packets received, 0%
packet loss
round-trip min/avg/max = 0.127/0.153/0.180 ms

c1host:/# curl winsweb

Hey, this is WINstagram!
Your IP Address is: 100.100.15.21
```

T1-core-only - configure “C-1” (c1rt) → validation

```
c1host:/# ping -c 2 nxweb

PING nxweb (100.90.55.55): 56 data bytes

64 bytes from 100.90.55.55: seq=0 ttl=58
time=0.183 ms
64 bytes from 100.90.55.55: seq=1 ttl=58
time=0.116 ms

--- nxweb ping statistics ---

2 packets transmitted, 2 packets received, 0%
packet loss
round-trip min/avg/max = 0.116/0.149/0.183 ms

c1host:/# curl nxweb

Welcome to Newspaper-X.
Your IP Address is: 100.100.15.21
```

WHY ???

```
c1host:/# ping -c 2 winsweb

PING winsweb (100.80.44.44): 56 data bytes

64 bytes from 100.80.44.44: seq=0 ttl=59
time=0.180 ms
64 bytes from 100.80.44.44: seq=1 ttl=59
time=0.127 ms

--- winsweb ping statistics ---

2 packets transmitted, 2 packets received, 0%
t loss
-trip min/avg/max = 0.127/0.153/0.180 ms

c1host:/# curl winsweb

Hey, this is WINstagram!
Your IP Address is: 100.100.15.21
```

T1-core-only - configure “C-1” (c1rt) → validation

How can we simulate a “link failure”?

netlab uses as backend a tool called “*containerlab*”, which allows to simulate packet loss on a link.

Let’s simulate a link loss of 100% on the link between ***c1rt*** and ***corp1***.

```
sudo containerlab tools netem set -n clab-T1-core-only-c1rt -i eth1 --loss 100
```

name of the virtual router / container:
clab-<topology_name>-<device_name>

… and let’s wait some seconds for failover to happen.

T1-core-only - configure “C-1” (c1rt) → validation

```
c1host:/# traceroute nxweb

traceroute to nxweb (100.90.55.55), 30 hops max, 46 byte packets

 1   100.100.15.20 (100.100.15.20)  0.010 ms  0.008 ms  0.009 ms
 2   100.100.2.6 (100.100.2.6)  0.008 ms  0.008 ms  0.009 ms
 3   100.100.1.37 (100.100.1.37)  0.008 ms  0.008 ms  0.008 ms
 4   100.100.1.6 (100.100.1.6)  0.008 ms  0.009 ms  0.008 ms
 5   100.71.0.2 (100.71.0.2)  0.008 ms  0.008 ms  0.008 ms
 6   100.71.0.9 (100.71.0.9)  0.008 ms  0.008 ms  0.008 ms
 7   nxweb (100.90.55.55)  0.008 ms  0.009 ms  0.008 ms
```

c1rt

!!! corp2 !!!

T2-empty-acme

- This topology creates all the virtual devices, all the interfaces and related addressing.
- It configures BGP on all the devices except ACME's.
- **It will allow you to configure on your own OSPF, BGP and related policies on the ACME devices**
- (login to any other device to check the routes, or use "`netlab validate`" **)

```
cd T2-empty-acme
netlab up
```

T2-empty-acme

Configuration steps:

1. Configure OSPF:
   a. internal links
   b. loopback interfaces
2. Configure iBGP peerings (RR and RR clients)
3. Configure eBGP policies
4. Configure eBGP peers

T2-empty-acme - OSPF example

corp1

```
interface eth1
 ip ospf area 0.0.0.0
 ip ospf network point-to-point

!

interface eth2
 ip ospf area 0.0.0.0
 ip ospf network point-to-point

!

interface lo
 ip ospf area 0.0.0.0
```

T2-empty-acme - iBGP RR example

core1

```
router bgp 64666
 bgp cluster-id 100.100.0.1
 neighbor 100.100.0.2 remote-as 64666
 neighbor 100.100.0.2 description core2
 neighbor 100.100.0.2 update-source lo
 neighbor 100.100.0.11 remote-as 64666
 neighbor 100.100.0.11 description igw1
 neighbor 100.100.0.11 update-source lo
 neighbor 100.100.0.12 remote-as 64666
 neighbor 100.100.0.12 description igw2
 neighbor 100.100.0.12 update-source lo
 neighbor 100.100.0.13 remote-as 64666
 neighbor 100.100.0.13 description ixp
 neighbor 100.100.0.13 update-source lo
 neighbor 100.100.0.14 remote-as 64666
 neighbor 100.100.0.14 description corp1
 neighbor 100.100.0.14 update-source lo
 neighbor 100.100.0.15 remote-as 64666
 neighbor 100.100.0.15 description corp2
 neighbor 100.100.0.15 update-source lo
 neighbor 100.100.0.16 remote-as 64666
 neighbor 100.100.0.16 description bras
 neighbor 100.100.0.16 update-source lo
```

```
address-family ipv4 unicast
 network 100.100.0.0/20
 network 100.100.0.1/32
 neighbor 100.100.0.2 activate
 neighbor 100.100.0.2 next-hop-self
 neighbor 100.100.0.11 activate
 neighbor 100.100.0.11 route-reflector-client
 neighbor 100.100.0.11 next-hop-self
 neighbor 100.100.0.12 activate
 neighbor 100.100.0.12 route-reflector-client
 neighbor 100.100.0.12 next-hop-self
 neighbor 100.100.0.13 activate
 neighbor 100.100.0.13 route-reflector-client
 neighbor 100.100.0.13 next-hop-self
 neighbor 100.100.0.14 activate
 neighbor 100.100.0.14 route-reflector-client
 neighbor 100.100.0.14 next-hop-self
 neighbor 100.100.0.15 activate
 neighbor 100.100.0.15 route-reflector-client
 neighbor 100.100.0.15 next-hop-self
 neighbor 100.100.0.16 activate
 neighbor 100.100.0.16 route-reflector-client
 neighbor 100.100.0.16 next-hop-self
exit-address-family
```

T2-empty-acme - iBGP RR Client example

corp1

```
router bgp 64666

 neighbor 100.100.0.1 remote-as 64666
 neighbor 100.100.0.1 description core1
 neighbor 100.100.0.1 update-source lo

 neighbor 100.100.0.2 remote-as 64666
 neighbor 100.100.0.2 description core2
 neighbor 100.100.0.2 update-source lo
```

```
address-family ipv4 unicast

 neighbor 100.100.0.1 activate
 neighbor 100.100.0.1 next-hop-self

 neighbor 100.100.0.2 activate
 neighbor 100.100.0.2 next-hop-self

exit-address-family
```

T2-empty-acme - eBGP policy example

```
ip prefix-list my_aggregate seq 1 permit 100.100.0.0/20

!

bgp as-path access-list 1 permit ^$

!

route-map bp-transit1-1-out permit 10
 match as-path 1
 match ip address prefix-list my_aggregate


!

route-map bp-transit1-1-out deny 99

!

route-map bp-transit1-1-in permit 10
 set local-preference 100
```

igw1

```
ip prefix-list my_aggregate seq 1 permit 100.100.0.0/20

!

bgp as-path access-list 1 permit ^$

!

route-map bp-transit2-1-out permit 10
 match as-path 1
 match ip address prefix-list my_aggregate
 set as-path prepend 64666

!

route-map bp-transit2-1-out deny 99

!

route-map bp-transit2-1-in permit 10
 set local-preference 50
```

igw2

T2-empty-acme - eBGP peering example

```
router bgp 64666

 neighbor 100.71.0.2 remote-as 65001
 neighbor 100.71.0.2 description transit1

 address-family ipv4 unicast

  neighbor 100.71.0.2 activate

  neighbor 100.71.0.2 route-map bp-transit1-1-in in
  neighbor 100.71.0.2 route-map bp-transit1-1-out out

 exit-address-family
```

igw1

```
router bgp 64666

 neighbor 100.72.0.2 remote-as 65002
 neighbor 100.72.0.2 description transit2

 address-family ipv4 unicast

  neighbor 100.72.0.2 activate

  neighbor 100.72.0.2 route-map bp-transit2-1-in in
  neighbor 100.72.0.2 route-map bp-transit2-1-out out

 exit-address-family
```

igw2

Let's check the status of BGP Routes on core1

focus on routes for Newspaper-X

```
core1# show ip bgp 100.90.55.0/24
BGP routing table entry for 100.90.55.0/24
Paths: (1 available, best #1, table default)
  Advertised to non peer-group peers:
  core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
  65001 65050, (Received from a RR-client)
    100.100.0.11(igw1) (metric 10) from igw1(100.100.0.11) (100.100.0.11)
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS 65001, best (First path received)
      Last update: Sun Jul 14 14:13:09 2024
```

I am receiving only one route, and only from IGW1. This is the best route, so it should be ok.
But… why IGW2 is not announcing it? It shall be receiving it via Transit-2…

→ **let's check IGW2.**

Let's check the status of BGP Routes on igw2

focus on routes for Newspaper-X

```
igw2# show ip bgp 100.90.55.0/24
BGP routing table entry for 100.90.55.0/24
Paths: (3 available, best #1, table default)
  Not advertised to any peer
  65001 65050
    100.100.0.11(core1) (metric 20) from core1(100.100.0.1) (100.100.0.11)
      Origin IGP, metric 0, localpref 100, valid, internal, bestpath-from-AS 65001, best (Local Pref)
      Originator: 100.100.0.11, Cluster list: 100.100.0.1
      Last update: Sun Jul 14 14:13:09 2024
  65001 65050
    100.100.0.11(core2) (metric 20) from core2(100.100.0.2) (100.100.0.11)
      Origin IGP, metric 0, localpref 100, valid, internal
      Originator: 100.100.0.11, Cluster list: 100.100.0.1
      Last update: Sun Jul 14 14:13:09 2024
  65002 65050
    100.72.0.2(transit2) from transit2(100.72.0.2) (100.72.2.2)
      Origin IGP, localpref 50, valid, external, bestpath-from-AS 65002
      Last update: Sun Jul 14 14:13:09 2024
```

"*not advertised to any peer*"

IGW2 is receiving a "best route" from core1 (and 2), so it is not announcing the route back.

But... If I need to announce the "full routes" to my peers? Or to improve my convergence?

I should receive it on my core.

caveat #1: add support for BGP multipath/addpath

The rules of BGP (*RFC 4271*) prevent sending back a received route/path. If a BGP speaker advertises two paths to the same prefix, the second update overwrites the first one because they describe the same prefix.

RFC 7911 extends the advertised prefix (*Network Layer Reachability Information – NLRI*) with a **Path Identifier** to solve the multiple updates of the same prefix. This can be explicitly configured on our routers.

In small IBGP network, the different routers can send additional paths to the neighbors to enable multipathing or optimal path selection.

Instead, in larger networks, you might want to reduce the number of additional paths sent to edge routers – it's always a tradeoff between memory/CPU utilization and path selection optimality.

Solution:

add, to the BGP address family ipv4 config:

```
maximum-paths 8
maximum-paths ibgp 8
```

add, to any iBGP peer (on address family ipv4):

```
neighbor 1.2.3.4 addpath-tx-all-paths
```

→ **let's try again.**

Let's check the status of BGP Routes on core1

focus on routes for Newspaper-X

```
core1# show ip bgp 100.90.55.0/24
BGP routing table entry for 100.90.55.0/24
Paths: (2 available, best #1, table default)
  65001 65050, (Received from a RR-client)
    100.100.0.11(igw1) (metric 10) from igw1(100.100.0.11) (100.100.0.11)
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS 65001, best (Local Pref)
      AddPath ID: RX 20, TX-All 24 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 14:40:27 2024
  65002 65050, (Received from a RR-client)
    100.100.0.12(igw2) (metric 10) from igw2(100.100.0.12) (100.100.0.12)
      Origin IGP, localpref 50, valid, internal, bestpath-from-AS 65002
      AddPath ID: RX 20, TX-All 25 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 14:40:27 2024
```

We like it! Now we have on our core routers all the different "routing options" (**paths**).
You can also see the "AddPath identifier" associated to each path.

Let's check the status of BGP Routes on core1

focus on routes for WINstagram

```
core1# show ip bgp 100.80.44.0/24
BGP routing table entry for 100.80.44.0/24
Paths: (3 available, best #1, table default)
  64777 65040, (Received from a RR-client)
    100.100.0.13(ixp) (metric 10) from ixp(100.100.0.13) (100.100.0.13)
      Origin IGP, localpref 200, valid, internal, bestpath-from-AS 64777, best (Local Pref)
      AddPath ID: RX 2, TX-All 12 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 14:48:33 2024
  65002 65040, (Received from a RR-client)
    100.100.0.12(igw2) (metric 10) from igw2(100.100.0.12) (100.100.0.12)
      Origin IGP, localpref 50, valid, internal, bestpath-from-AS 65002
      AddPath ID: RX 44, TX-All 26 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 14:48:33 2024
  65001 65040, (Received from a RR-client)
    100.100.0.11(igw1) (metric 10) from igw1(100.100.0.11) (100.100.0.11)
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS 65001
      AddPath ID: RX 44, TX-All 27 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 14:48:33 2024
```

WAIT! The path to WINstagram via IXP is not what we were expecting.

A Route Server shall not add itself to the path!!! Here, instead, we see it: AS 64777. This will make the path longer!

→ **Let's check on IXP router.**

Let's check the status of BGP Routes on ixp

focus on routes for WINstagram

```
ixp# show ip bgp 100.80.44.0/24
BGP routing table entry for 100.80.44.0/24
Paths: (5 available, best #1, table default)
  64777 65040
    100.70.70.40(ixp_rs) from ixp_rs(100.70.70.70) (100.70.71.72)
      Origin IGP, localpref 200, valid, external, bestpath-from-AS 64777, best (Local Pref)
      AddPath ID: RX 0, TX-All 2 TX-Best-Per-AS 0
      Advertised to: core1(100.100.0.1) core2(100.100.0.2)
      Last update: Sun Jul 14 14:48:32 2024
  65001 65040
    100.100.0.11(core2) (metric 20) from core2(100.100.0.2) (100.100.0.11)
```

WAIT! The path to WINstagram via IXP is not what we were expecting.

A Route Server shall not add itself to the path!!! Here, instead, we see it: AS 64777. This will make the path longer!

caveat #2: route server @ IXP

The IXP Router Server shall be properly configured to:

- do not insert its own AS in the AS-Path
- do not change the IP next-hop

This can be done with the following neighbor config:

```
neighbor 1.2.3.4 route-server-client
neighbor 1.2.3.4 attribute-unchanged next-hop
```

→ **let's try again.**

Let’s check the status of BGP Routes on core1

focus on routes for WINstagram

```
core1# show ip bgp 100.80.44.0/24
BGP routing table entry for 100.80.44.0/24
Paths: (3 available, best #1, table default)
  65040, (Received from a RR-client)
    100.100.0.13(ixp) (metric 10) from ixp(100.100.0.13) (100.100.0.13)
      Origin IGP, metric 20, localpref 200, valid, internal, bestpath-from-AS 65040, best (Local Pref)
      AddPath ID: RX 2, TX-All 12 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 14:57:55 2024
  65002 65040, (Received from a RR-client)
    100.100.0.12(igw2) (metric 10) from igw2(100.100.0.12) (100.100.0.12)
      Origin IGP, localpref 50, valid, internal, bestpath-from-AS 65002
      AddPath ID: RX 44, TX-All 26 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 14:57:55 2024
  65001 65040, (Received from a RR-client)
    100.100.0.11(igw1) (metric 10) from igw1(100.100.0.11) (100.100.0.11)
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS 65001
      AddPath ID: RX 44, TX-All 27 TX-Best-Per-AS 0
      Advertised to: core2(100.100.0.2) igw1(100.100.0.11) igw2(100.100.0.12) ixp(100.100.0.13) corp1(100.100.0.14) corp2(100.100.0.15) bras(100.100.0.16)
      Last update: Sun Jul 14 14:57:55 2024
```

We like it! Now we have the shortest AS-Path to WINstagram via the IXP link.

Topology validation

As we did before, you can validate the topology in multiple ways:

- show ip bgp [<prefix>] (on different routers)
- netlab validate
- traceroute from *c1host*, *c2host or c3host*
- simple reachability check (ping, curl)

Simple traceroute

```
c3host:/# traceroute winsweb
traceroute to winsweb (100.80.44.44), 30 hops max, 46 byte packets
 1  100.100.3.16 (100.100.3.16)  0.009 ms  0.008 ms  0.007 ms
 2  100.100.1.50 (100.100.1.50)  0.008 ms  0.008 ms  0.008 ms
 3  100.100.1.22 (100.100.1.22)  0.007 ms  0.008 ms  0.008 ms
 4  100.70.70.40 (100.70.70.40)  0.007 ms  0.009 ms  0.008 ms
 5  winsweb (100.80.44.44)  0.008 ms  0.008 ms  0.014 ms
```

Notice the shorter path to reach WINstagram w/ respect to Newspaper-X:
We are using the IXP peering!

```
c3host:/# traceroute nxweb
traceroute to nxweb (100.90.55.55), 30 hops max, 46 byte packets
 1  100.100.3.16 (100.100.3.16)  0.008 ms  0.008 ms  0.008 ms
 2  100.100.1.50 (100.100.1.50)  0.008 ms  0.008 ms  0.008 ms
 3  100.100.1.6 (100.100.1.6)  0.008 ms  0.008 ms  0.007 ms
 4  100.71.0.2 (100.71.0.2)  0.008 ms  0.008 ms  0.008 ms
 5  100.71.0.9 (100.71.0.9)  0.008 ms  0.008 ms  0.007 ms
 6  nxweb (100.90.55.55)  0.008 ms  0.008 ms  0.008 ms
```

New failover scenario...

Let's fully remove the BGP configuration both on ACME IXP router and ACME IGW1 router.

(do a config backup before doing this exercise!)

→ **What should we expect for reaching WINstagram?**

New failover scenario...

Before the failure: **we are using the IXP peering.**

```
c3host:/# traceroute winsweb
traceroute to winsweb (100.80.44.44), 30 hops max, 46 byte packets
 1  100.100.3.16 (100.100.3.16)  0.009 ms  0.008 ms  0.007 ms
 2  100.100.1.50 (100.100.1.50)  0.008 ms  0.008 ms  0.008 ms
 3  100.100.1.22 (100.100.1.22)  0.007 ms  0.008 ms  0.008 ms
 4  100.70.70.40 (100.70.70.40)  0.007 ms  0.009 ms  0.008 ms
 5  winsweb (100.80.44.44)  0.008 ms  0.008 ms  0.014 ms
```

After the failure: **we are using the Transit-2 link.**
(one additional hop)

```
c3host:/# traceroute winsweb
traceroute to winsweb (100.80.44.44), 30 hops max, 46 byte packets
 1  100.100.3.16 (100.100.3.16)  0.010 ms  0.008 ms  0.008 ms
 2  100.100.1.50 (100.100.1.50)  0.009 ms  0.008 ms  0.008 ms
 3  100.100.1.14 (100.100.1.14)  0.008 ms  0.009 ms  0.009 ms
 4  100.72.0.2 (100.72.0.2)  0.008 ms  0.009 ms  0.008 ms
 5  100.72.0.6 (100.72.0.6)  0.008 ms  0.009 ms  0.008 ms
 6  winsweb (100.80.44.44)  0.009 ms  0.009 ms  0.024 ms
```

New failover scenario…

```
[aspath_t1]        Check correct AS-Path on Transit-1 [ node(s): transit1 ]
[FAIL]             transit1: The prefix 100.100.0.0/20 is not in the BGP table

[aspath_t2]        Check correct AS-Path on Transit-2 [ node(s): transit2 ]
[PASS]             transit2: The prefix 100.100.0.0/20 is in the BGP table with AS path=64666 64666 64666
[PASS]             Test succeeded

[c1_bestpath]      Check correct Best Path on Customer-1 Router [ node(s): c1rt ]
[PASS]             c1rt: The prefix 100.100.0.0/20 is in the BGP table with BGP router ID=100.100.0.14,best path=True
[PASS]             Test succeeded

[best_to_wins]     Check correct Best Path TO WINstagram [ node(s): core1 ]
[FAIL]             core1: There is no path to 100.80.44.0/24 in the BGP table with BGP router ID=100.100.0.13

[best_from_wins]   Check correct Best Path FROM WINstagram [ node(s): winsrt ]
[FAIL]             winsrt: The next hop(s) for prefix 100.100.0.0/20 is/are 100.72.0.5, not 100.70.70.13

[best_to_nx]       Check correct Best Path TO Newspaper-X [ node(s): core1 ]
[FAIL]             core1: There is no path to 100.90.55.0/24 in the BGP table with BGP router ID=100.100.0.11

[best_from_nx]     Check correct Best Path FROM Newspaper-X [ node(s): nxrt ]
[FAIL]             nxrt: There is no path to 100.100.0.0/20 in the BGP table with BGP router ID=100.71.1.1

[ping_wins]        Pinging WINstagram Web Server [ node(s): c1host,c2host,c3host ]
[PASS]             Validation succeeded on c1host
[PASS]             Validation succeeded on c2host
[PASS]             Validation succeeded on c3host
[PASS]             Test succeeded

[ping_nx]          Pinging Newspaper-X Web Server [ node(s): c1host,c2host,c3host ]
[PASS]             Validation succeeded on c1host
[PASS]             Validation succeeded on c2host
[PASS]             Validation succeeded on c3host
[PASS]             Test succeeded
```

Errors on lab validation…

But reachability is still guaranteed!

BONUS: packet capture from the codespace

From the codespace, it's possible to launch an instance of "Edgeshark": a Web UI that displays every interface of every container (virtual device) and can start a wireshark session by a click of a button.

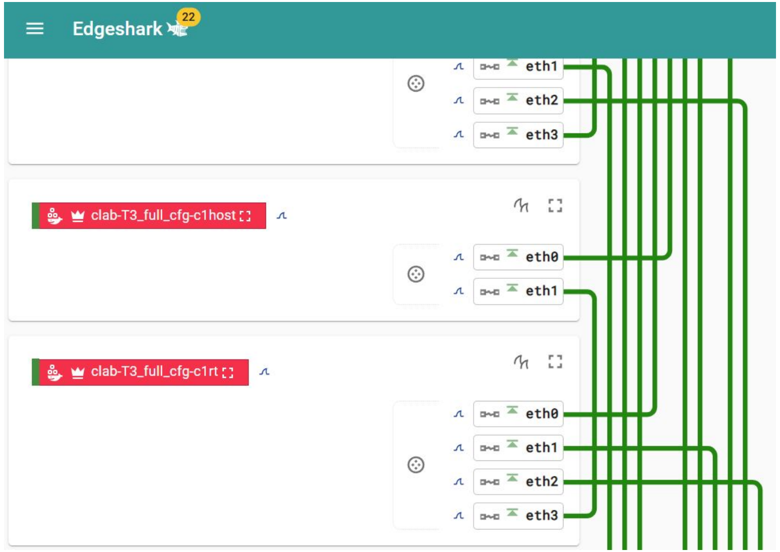
NOTE: Wireshark integration requires:

- Wireshark installed on your computer
- the installation of Edgeshark agent

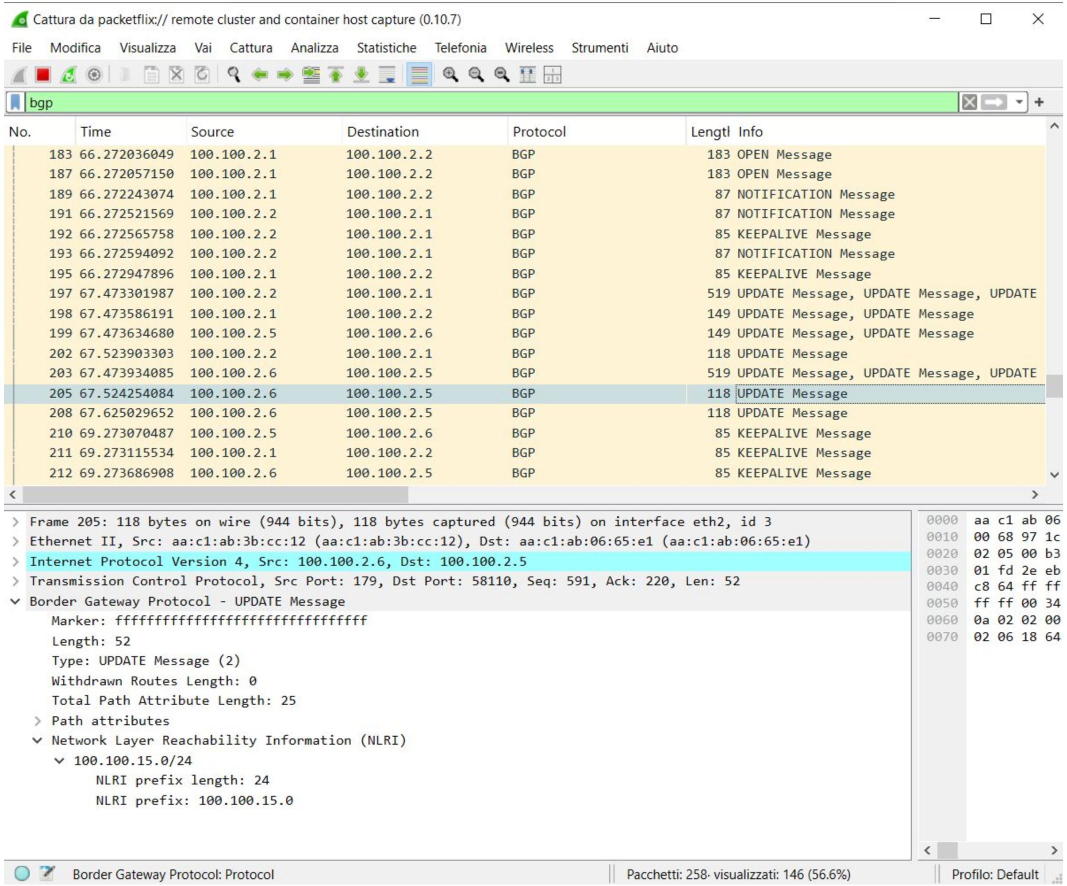
Look at the "**_edgeshark**" directory in the repository for a quickstart guide.

Ref: https://github.com/ssasso/acme-isp/blob/master/_edgeshark/README.md

Edgeshark



Edgeshark
22
eth1
eth2
eth3
clab-T3_full_cfg-c1host
eth0
eth1
clab-T3_full_cfg-c1rt
eth0
eth1
eth2
eth3



Cattura da packetflix:// remote cluster and container host capture (0.10.7)
File Modifica Visualizza Vai Cattura Analizza Statistiche Telefonia Wireless Strumenti Aiuto
bgp
No. Time Source Destination Protocol Length Info
183 66.272036049 100.100.2.1 100.100.2.2 BGP 183 OPEN Message
187 66.272057150 100.100.2.1 100.100.2.2 BGP 183 OPEN Message
189 66.272243074 100.100.2.1 100.100.2.2 BGP 87 NOTIFICATION Message
191 66.272521569 100.100.2.2 100.100.2.1 BGP 87 NOTIFICATION Message
192 66.272565758 100.100.2.2 100.100.2.1 BGP 85 KEEPALIVE Message
193 66.272594092 100.100.2.2 100.100.2.1 BGP 87 NOTIFICATION Message
195 66.272947896 100.100.2.1 100.100.2.2 BGP 85 KEEPALIVE Message
197 67.473301987 100.100.2.2 100.100.2.1 BGP 519 UPDATE Message, UPDATE Message, UPDATE
198 67.473586191 100.100.2.1 100.100.2.2 BGP 149 UPDATE Message, UPDATE Message
199 67.473634680 100.100.2.5 100.100.2.6 BGP 149 UPDATE Message, UPDATE Message
202 67.523903303 100.100.2.2 100.100.2.1 BGP 118 UPDATE Message
203 67.473934085 100.100.2.6 100.100.2.5 BGP 519 UPDATE Message, UPDATE Message, UPDATE
205 67.524254084 100.100.2.6 100.100.2.5 BGP 118 UPDATE Message
208 67.625029652 100.100.2.6 100.100.2.5 BGP 118 UPDATE Message
210 69.273070487 100.100.2.5 100.100.2.6 BGP 85 KEEPALIVE Message
211 69.273115534 100.100.2.1 100.100.2.2 BGP 85 KEEPALIVE Message
212 69.273686908 100.100.2.6 100.100.2.5 BGP 85 KEEPALIVE Message
Frame 205: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface eth2, id 3
Ethernet II, Src: aa:c1:ab:3b:cc:12 (aa:c1:ab:3b:cc:12), Dst: aa:c1:ab:06:65:e1 (aa:c1:ab:06:65:e1)
Internet Protocol Version 4, Src: 100.100.2.6, Dst: 100.100.2.5
Transmission Control Protocol, Src Port: 179, Dst Port: 58110, Seq: 591, Ack: 220, Len: 52
Border Gateway Protocol - UPDATE Message
Marker: ffffffffffffffffffffffffffffffff
Length: 52
Type: UPDATE Message (2)
Withdrawn Routes Length: 0
Total Path Attribute Length: 25
Path attributes
Network Layer Reachability Information (NLRI)
100.100.15.0/24
NLRI prefix length: 24
NLRI prefix: 100.100.15.0
0000 aa c1 ab 06
0010 00 68 97 1c
0020 02 05 00 b3
0030 01 fd 2e eb
0040 c8 64 ff ff
0050 ff ff 00 34
0060 0a 02 02 00
0070 02 06 18 64
Border Gateway Protocol: Protocol
Pacchetti: 258· visualizzati: 146 (56.6%)
Profilo: Default

want more?

https://bgplabs.net/