IPDC

Your Trusted Cybersecurity Partner

Netflow Data Analytics With ELK Stack & DDoS Attack Mitigation

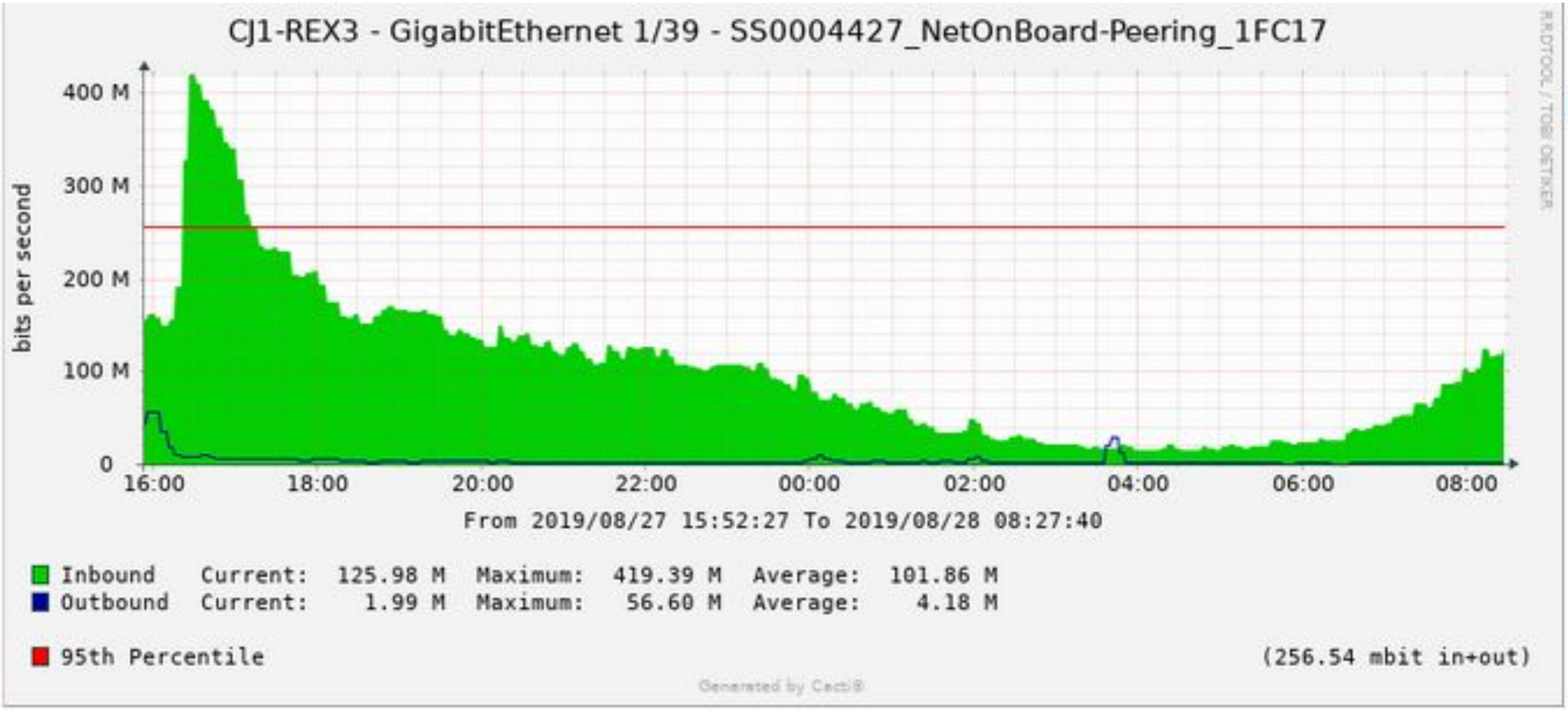
About
IPDC SOLUTIONS

- Founded in **2016**
- Over **15** employees
- Managing 700Gbit/s DDOS Mitigation capacity in **MY, SG, HK, TW, US** and **EU** on the way
- Providing DDOS protection solution for more than **100** ISPs
- Development on DDOS & Traffic monitoring system - **INI**

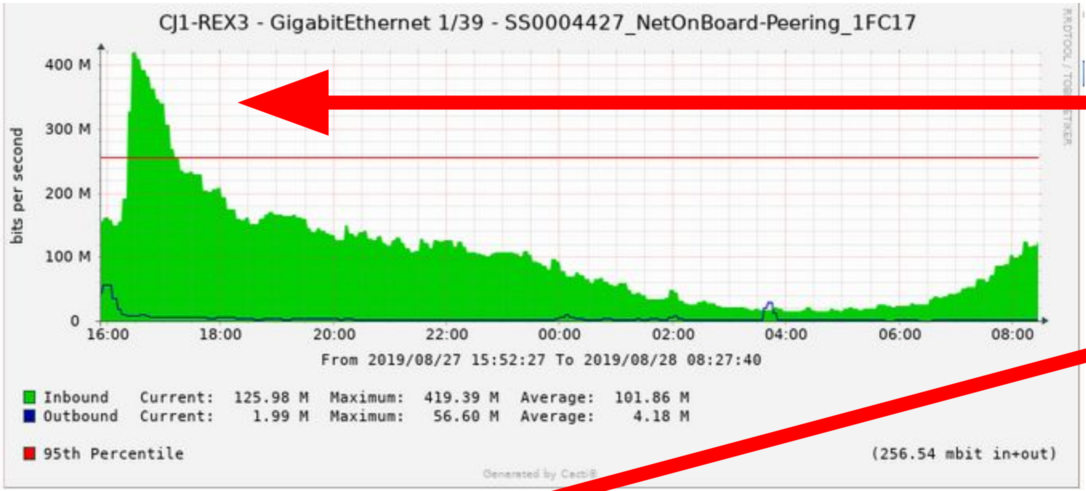
Why do we need to develop our own **NetFlow tools?**

As We need to resolve some operation difficulties, that required information that cannot be found from MRTG

Example 1. When we see a spike like the following graph



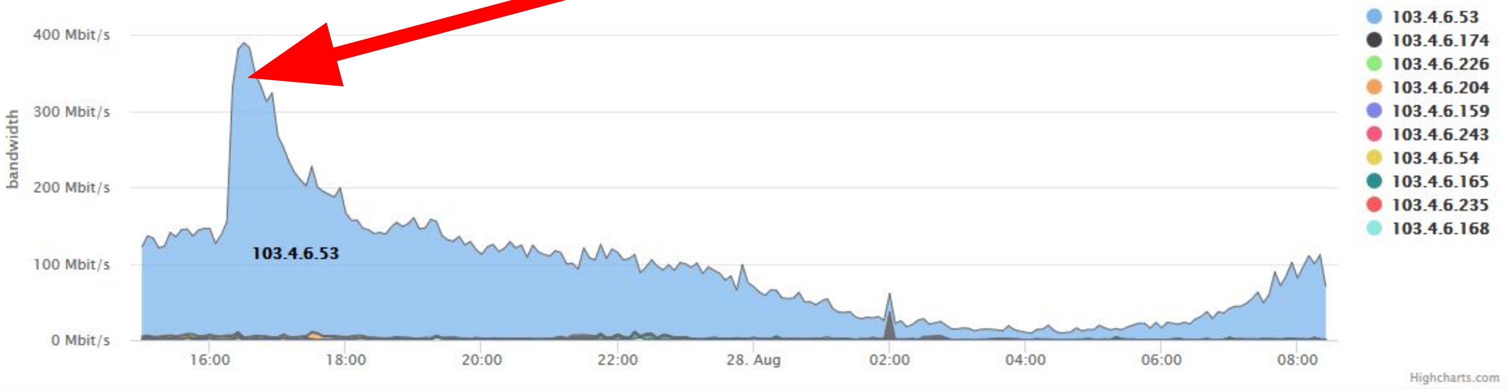
You probably may need to know where the majority of your traffic comes from?



```
From 2019/08/27 15:52:27 To 2019/08/28 08:27:40

🟩 Inbound   Current:   125.98 M  Maximum:  419.39 M  Average:  101.86 M
■ Outbound  Current:     1.99 M  Maximum:   56.60 M  Average:    4.18 M

🟥 95th Percentile                                         (256.54 mbit in+out)
```

Who uses the most bandwidth here?

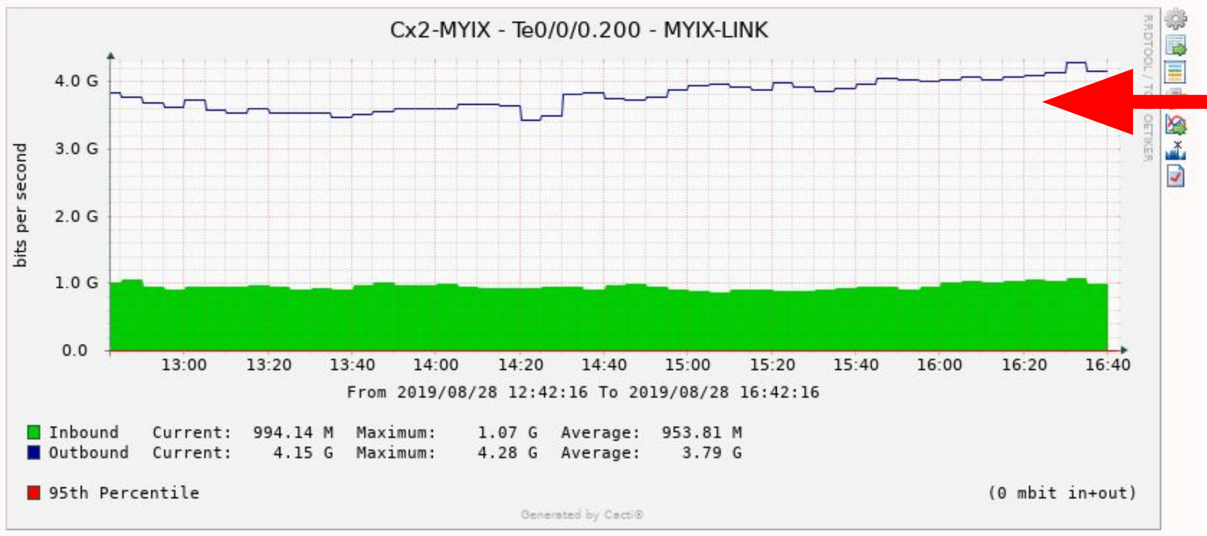
We found this from Netflow graph



#	IP Address	Current	Max	Avg	Min	95%
1	**103.4.6.53**	124.02	385.87	90.34	8.05	223.72
2	**103.4.6.174**	1.26	35.81	0.70	0.00	2.70
3	**103.4.6.226**	0.56	3.24	0.63	0.04	1.09
4	**103.4.6.204**	1.48	7.74	0.54	0.00	2.20

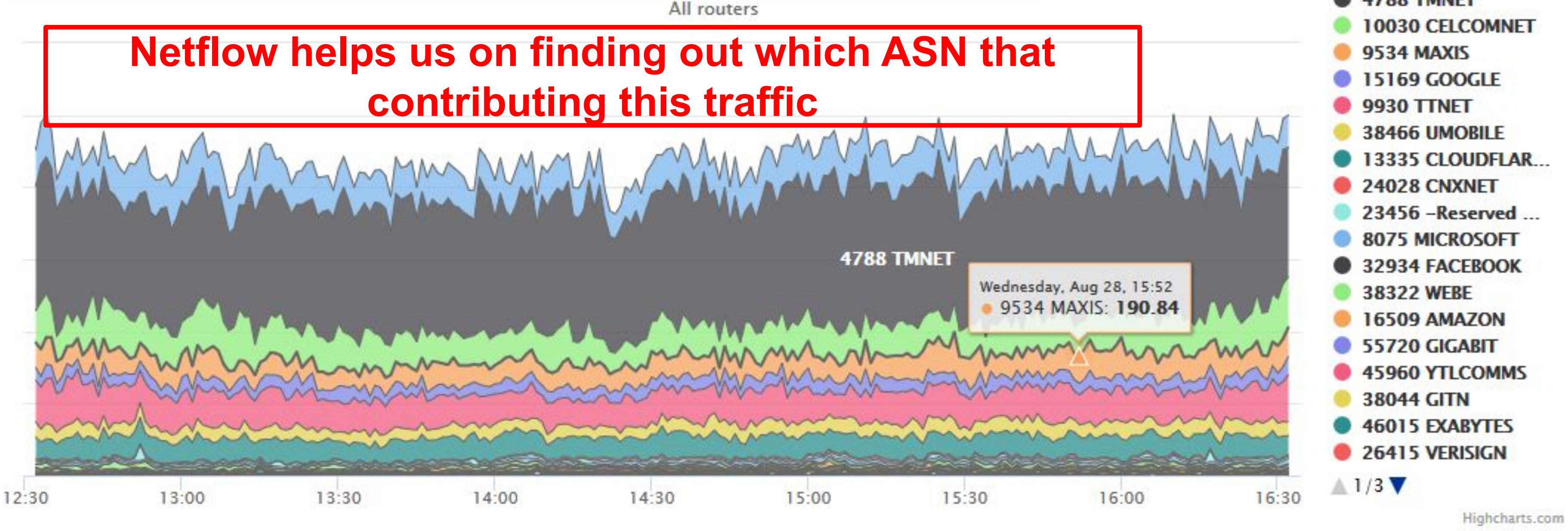
A NetFlow graph would be able to breakdown the usage

for your outbound / inbound traffic



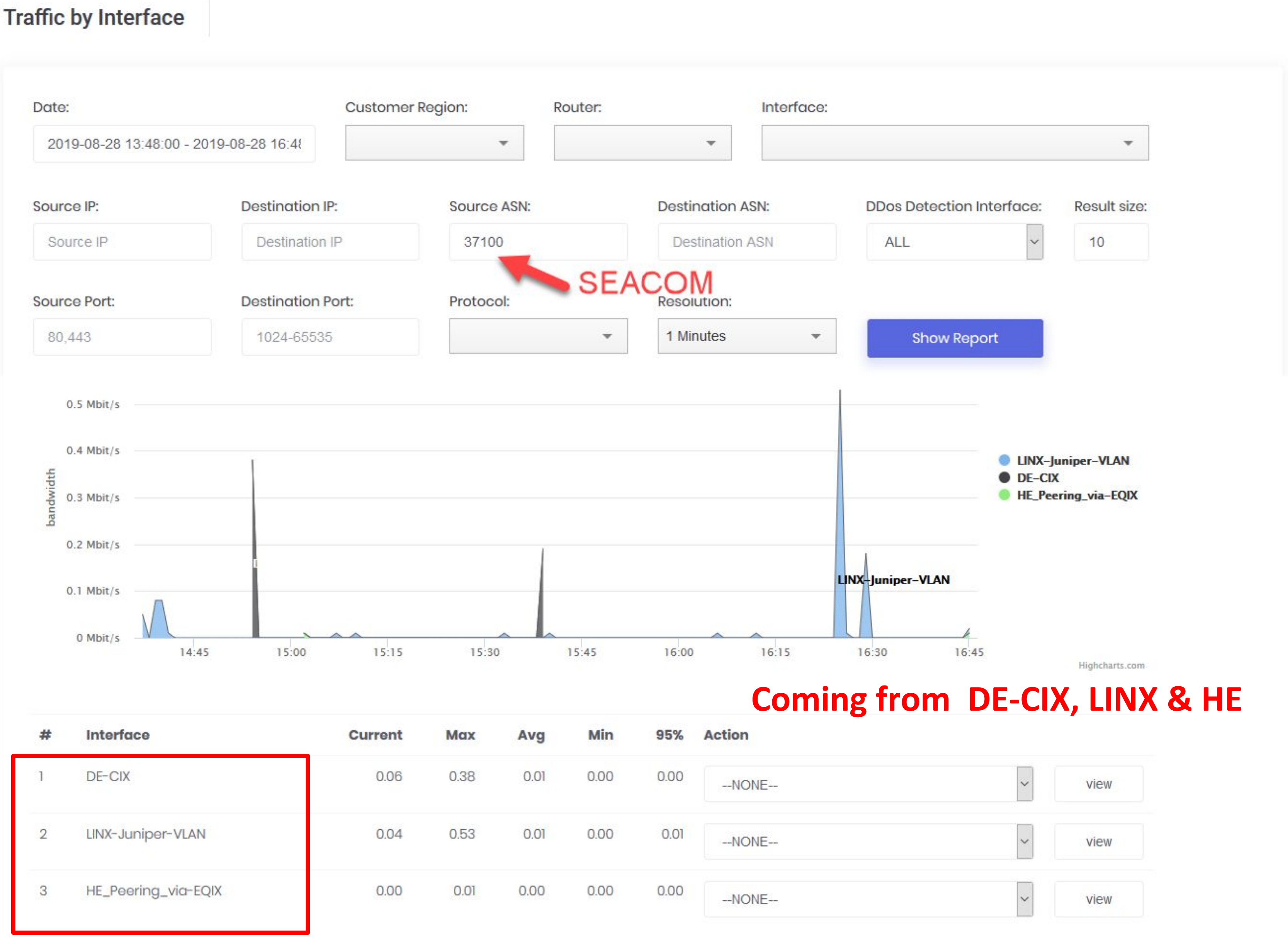
```
From 2019/08/28 12:42:16 To 2019/08/28 16:42:16

🟩 Inbound   Current:  994.14 M  Maximum:   1.07 G  Average:  953.81 M
🟦 Outbound  Current:    4.15 G  Maximum:   4.28 G  Average:    3.79 G

🟥 95th Percentile                                        (0 mbit in+out)
```

When we almost hitting your committed bandwidth limit.



Netflow helps us on finding out which ASN that contributing this traffic

Netflow is not just for graphing purposes.
It helps on how identify, which upstream / interface the traffic coming from



Traffic by Interface

Date: 2019-08-28 13:48:00 - 2019-08-28 16:4ε

Customer Region:

Router:

Interface:

Source IP: Source IP

Destination IP: Destination IP

Source ASN: 37100

Destination ASN: Destination ASN

DDos Detection Interface: ALL

Result size: 10

Source Port: 80,443

Destination Port: 1024-65535

Protocol:

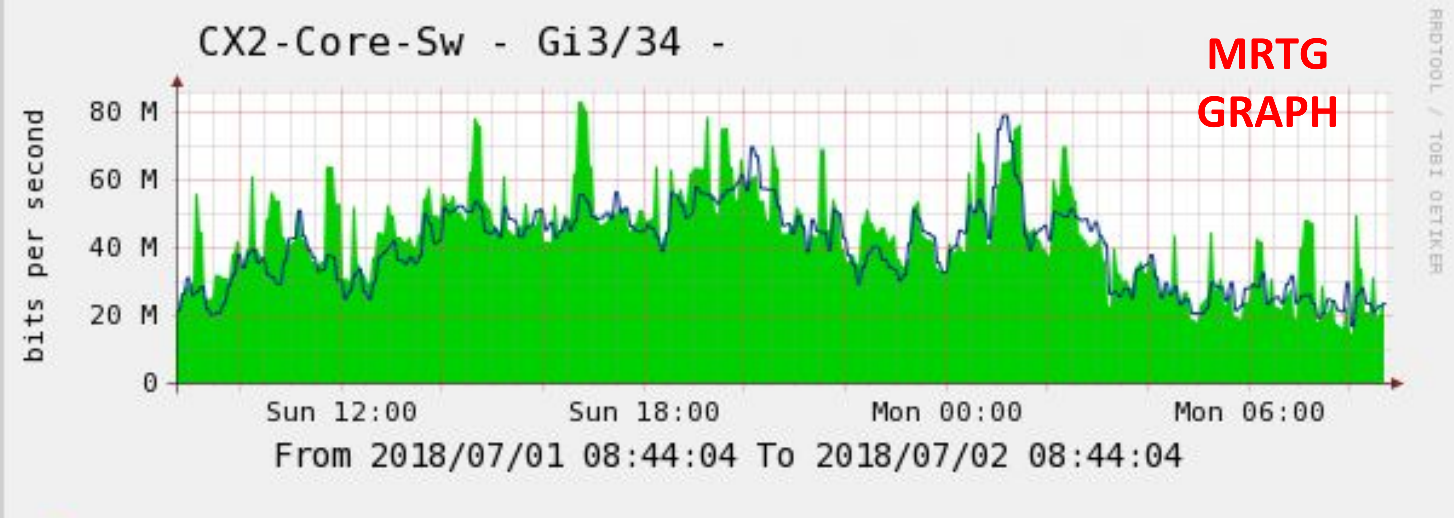
Resolution: 1 Minutes

SEACOM

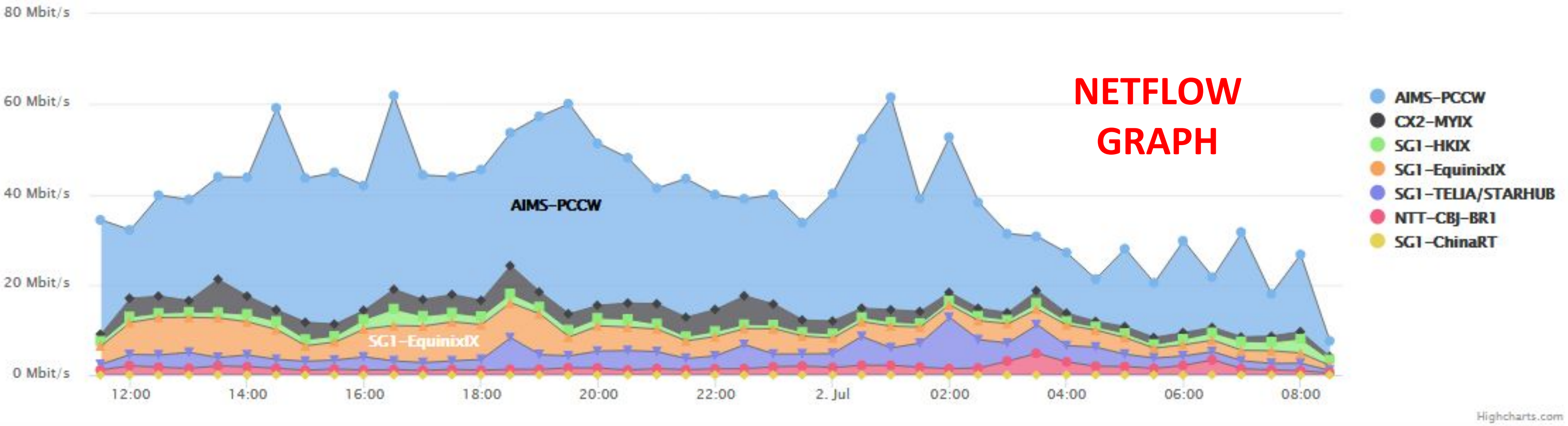
Coming from DE-CIX, LINX & HE

#	Interface	Current	Max	Avg	Min	95%	Action	
1	DE-CIX	0.06	0.38	0.01	0.00	0.00	--NONE--	view
2	LINX-Juniper-VLAN	0.04	0.53	0.01	0.00	0.01	--NONE--	view
3	HE_Peering_via-EQIX	0.00	0.01	0.00	0.00	0.00	--NONE--	view

If you considering to enhance your MRTG with a NetFlow graph
Here is how we do…



MRTG GRAPH



NETFLOW GRAPH

To get the graph plotted, we will need to store them into a database

ElasticSearch, Logstash, And Kibana (ELK)?

Why ELK?

- Before I get to know ELK stack, I was using MySQL to store all the NetFlow information.
- I wrote a PHP application that converts NetFlow information into a MySQL statement.
- That was too slow on the conversion performance and the data retrieval was a complete nightmare.
- There is no function / feature to get traffic statistic in the histogram form.

It's just too difficult to run this in MySQL

Why ELK?

- Speed is the primary reason that I have chosen ELK
- It has a lot of codec, which I can just plug and play
- COST; it runs on commodity hardware and it works just fine with Nearline SAS Hard drives
- Open Source
- Support Clustering
- It has SQL like syntax, so data searching is much more easier
- It has a very high performance; we had a working environment of 100Kflows per second

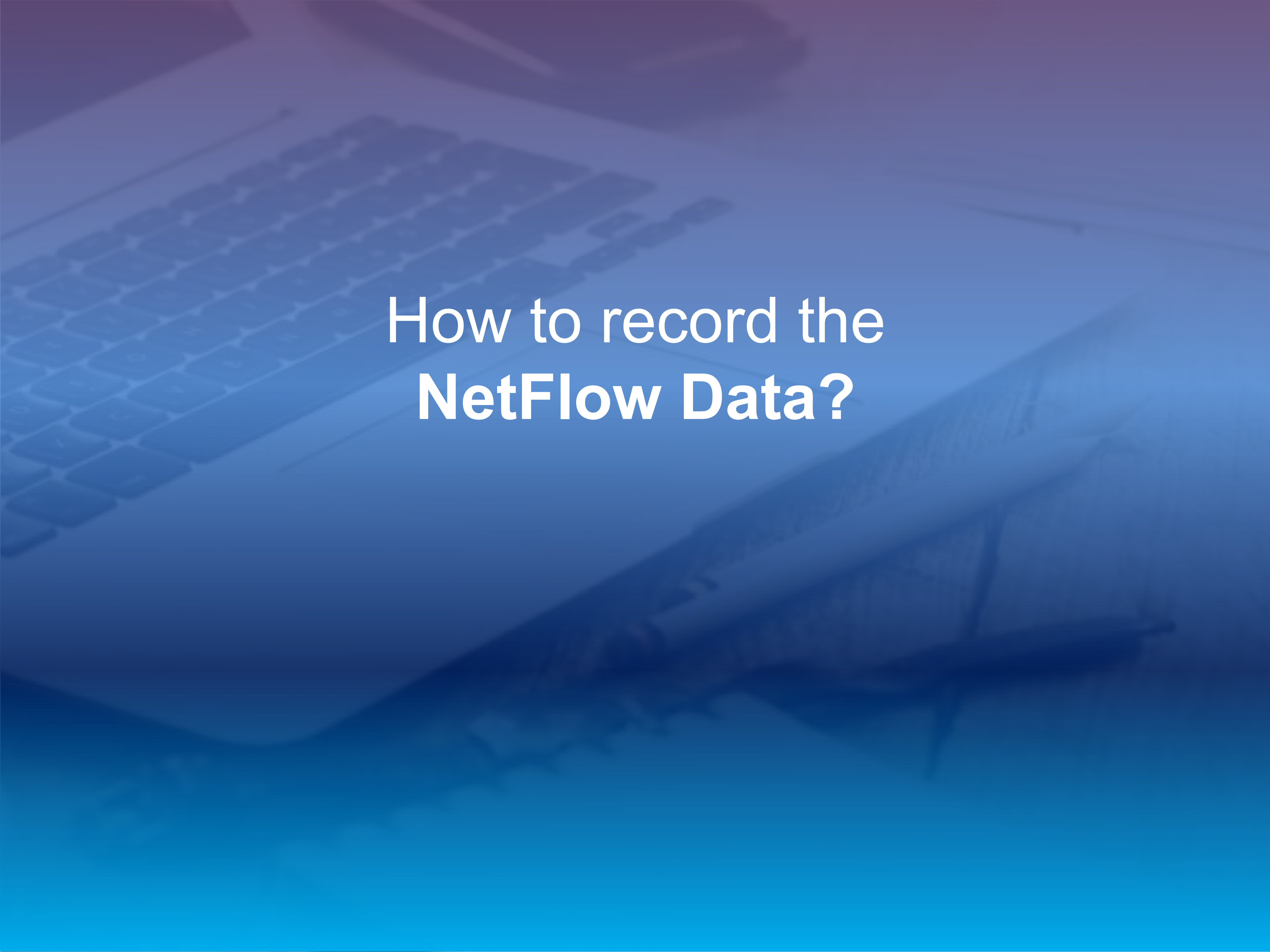
Alternative to ELK

- **We did consider to use InfluxDB**
  The OpenSource edition doesn't support clustering.

- **OpenTSDB**
  The setup is very time-consuming.

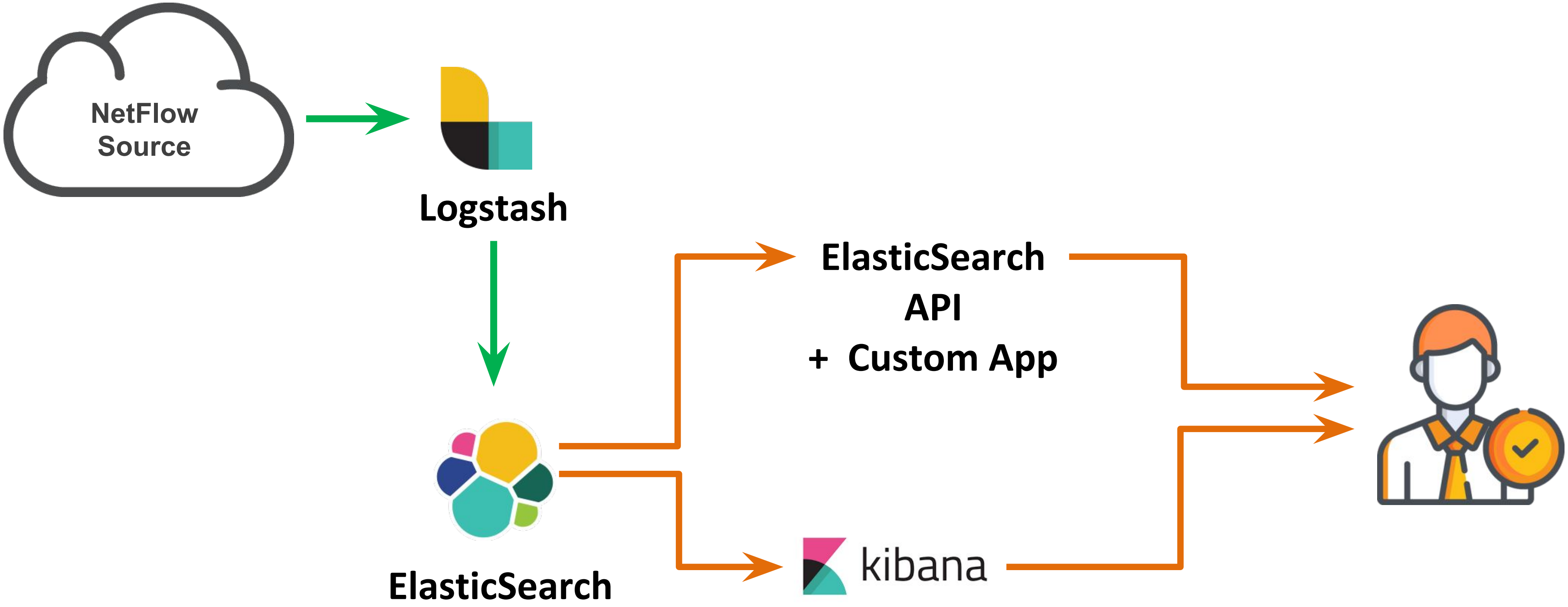
- **MongoDB.**
  This is a great DB; however, we still prefer to use ElasticSearch.

- **ClickHouse**
  ClickHouse is an open-source column-oriented DBMS for online analytical processing. ClickHouse was developed by the Russian IT company Yandex



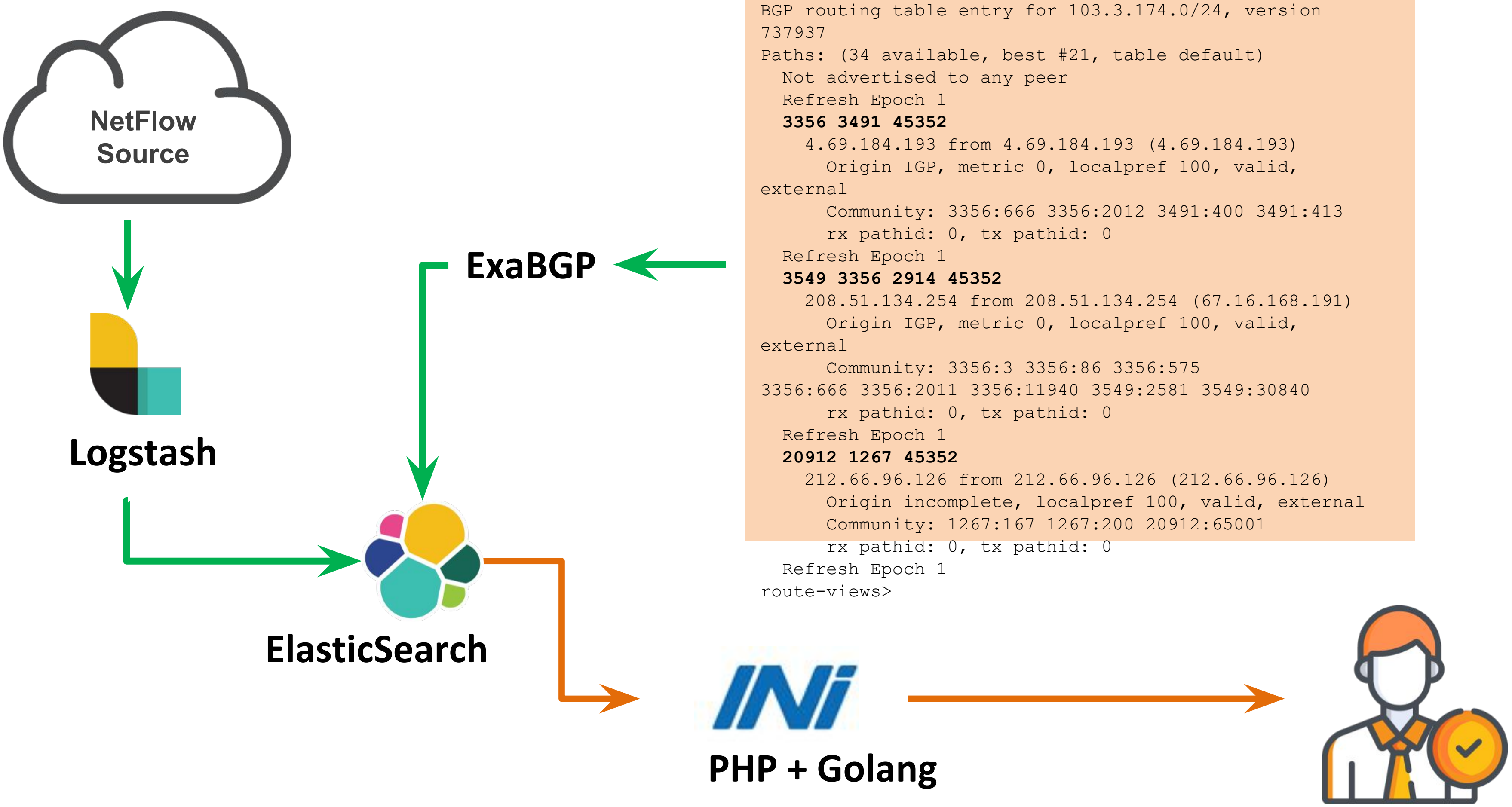
How to record the **NetFlow Data?**

The NetFlow is being collected with the following setup



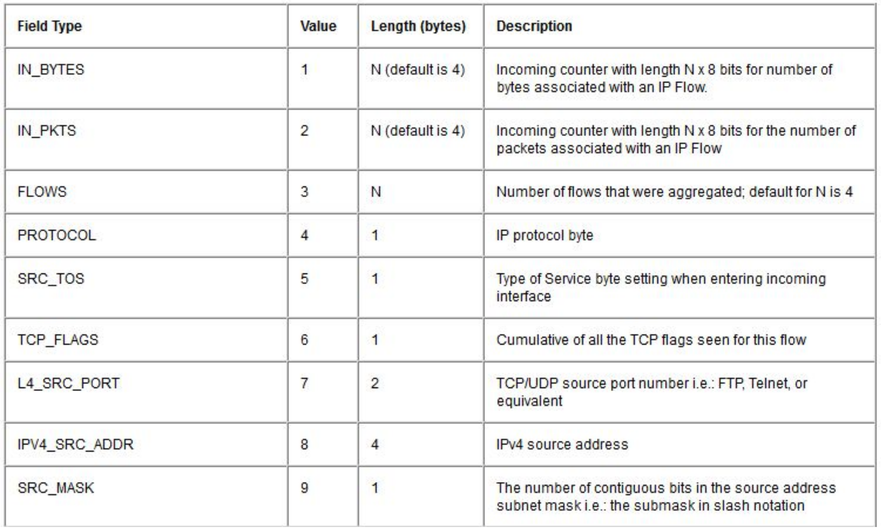
Adding BGP table information into the ElasticSearch

BGP Routing Table

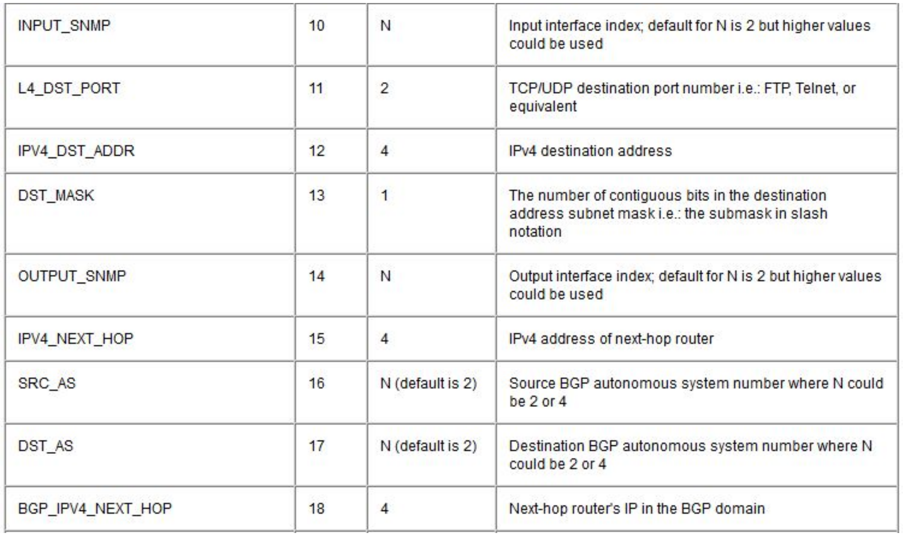


```
BGP routing table entry for 103.3.174.0/24, version
737937
Paths: (34 available, best #21, table default)
  Not advertised to any peer
  Refresh Epoch 1
  3356 3491 45352
    4.69.184.193 from 4.69.184.193 (4.69.184.193)
      Origin IGP, metric 0, localpref 100, valid,
external
      Community: 3356:666 3356:2012 3491:400 3491:413
      rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  3549 3356 2914 45352
    208.51.134.254 from 208.51.134.254 (67.16.168.191)
      Origin IGP, metric 0, localpref 100, valid,
external
      Community: 3356:3 3356:86 3356:575
3356:666 3356:2011 3356:11940 3549:2581 3549:30840
      rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  20912 1267 45352
    212.66.96.126 from 212.66.96.126 (212.66.96.126)
      Origin incomplete, localpref 100, valid, external
      Community: 1267:167 1267:200 20912:65001
      rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
route-views>
```

We use NetFlow v9 in our projects
Here is the field that we keep



Field Type	Value	Length (bytes)	Description
IN_BYTES	1	N (default is 4)	Incoming counter with length N x 8 bits for number of bytes associated with an IP Flow.
IN_PKTS	2	N (default is 4)	Incoming counter with length N x 8 bits for the number of packets associated with an IP Flow
FLOWS	3	N	Number of flows that were aggregated; default for N is 4
PROTOCOL	4	1	IP protocol byte
SRC_TOS	5	1	Type of Service byte setting when entering incoming interface
TCP_FLAGS	6	1	Cumulative of all the TCP flags seen for this flow
L4_SRC_PORT	7	2	TCP/UDP source port number i.e.: FTP, Telnet, or equivalent
IPV4_SRC_ADDR	8	4	IPv4 source address
SRC_MASK	9	1	The number of contiguous bits in the source address subnet mask i.e.: the submask in slash notation



INPUT_SNMP	10	N	Input interface index; default for N is 2 but higher values could be used
L4_DST_PORT	11	2	TCP/UDP destination port number i.e.: FTP, Telnet, or equivalent
IPV4_DST_ADDR	12	4	IPv4 destination address
DST_MASK	13	1	The number of contiguous bits in the destination address subnet mask i.e.: the submask in slash notation
OUTPUT_SNMP	14	N	Output interface index; default for N is 2 but higher values could be used
IPV4_NEXT_HOP	15	4	IPv4 address of next-hop router
SRC_AS	16	N (default is 2)	Source BGP autonomous system number where N could be 2 or 4
DST_AS	17	N (default is 2)	Destination BGP autonomous system number where N could be 2 or 4
BGP_IPV4_NEXT_HOP	18	4	Next-hop router's IP in the BGP domain

The NetFlow is being collected with the following setup

The **hardware specification** used for keeping our NetFlow

- **1 x** Intel Xeon 8 cores **2.1Ghz** Processor
- **4 x 2TB** HDD
- **32GB** RAM
- **1 x** Gigabit Network Card

The **software** used to run our NetFlow

- **CentOS 7**
  64bit Operating System
- **Java**
- **MySQL**
- **PHP**
- **ElasticSearch, Logstash**

How to put up the software?

CentOS Installation

You can follow the way you do normally; but please remember to keep most of the free space into /var.

Quick intro about ElasticSearch

ElasticSearch is a search engine based on Lucene. It provides a distributed architecture, support multi-tenancy and full-text search engine with an HTTP web interface.

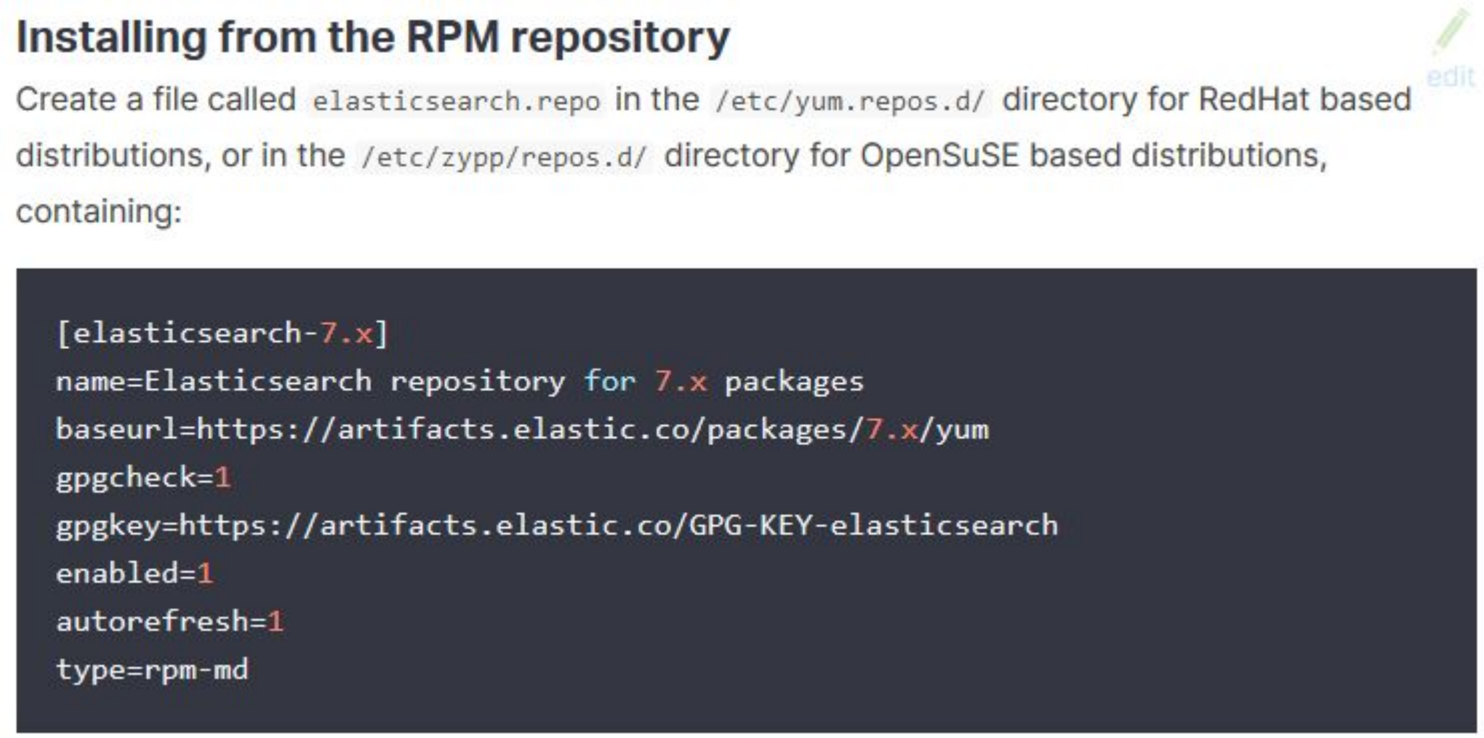
https://www.elastic.co/guide/en/elasticsearch/reference/current/rpm.html

Elasticsearch could support horizontal scaling
Where you cluster multiple server, and make it into 1 single cluster

To improve the processing capabilities, and also the speed to deliver your search result.

ElasticSearch Installation

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```



Installing from the RPM repository

Create a file called `elasticsearch.repo` in the `/etc/yum.repos.d/` directory for RedHat based distributions, or in the `/etc/zypp/repos.d/` directory for OpenSuSE based distributions, containing:

```
[elasticsearch-7.x]
name=Elasticsearch repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

```
sudo yum install elasticsearch
```

Start ElasticSearch

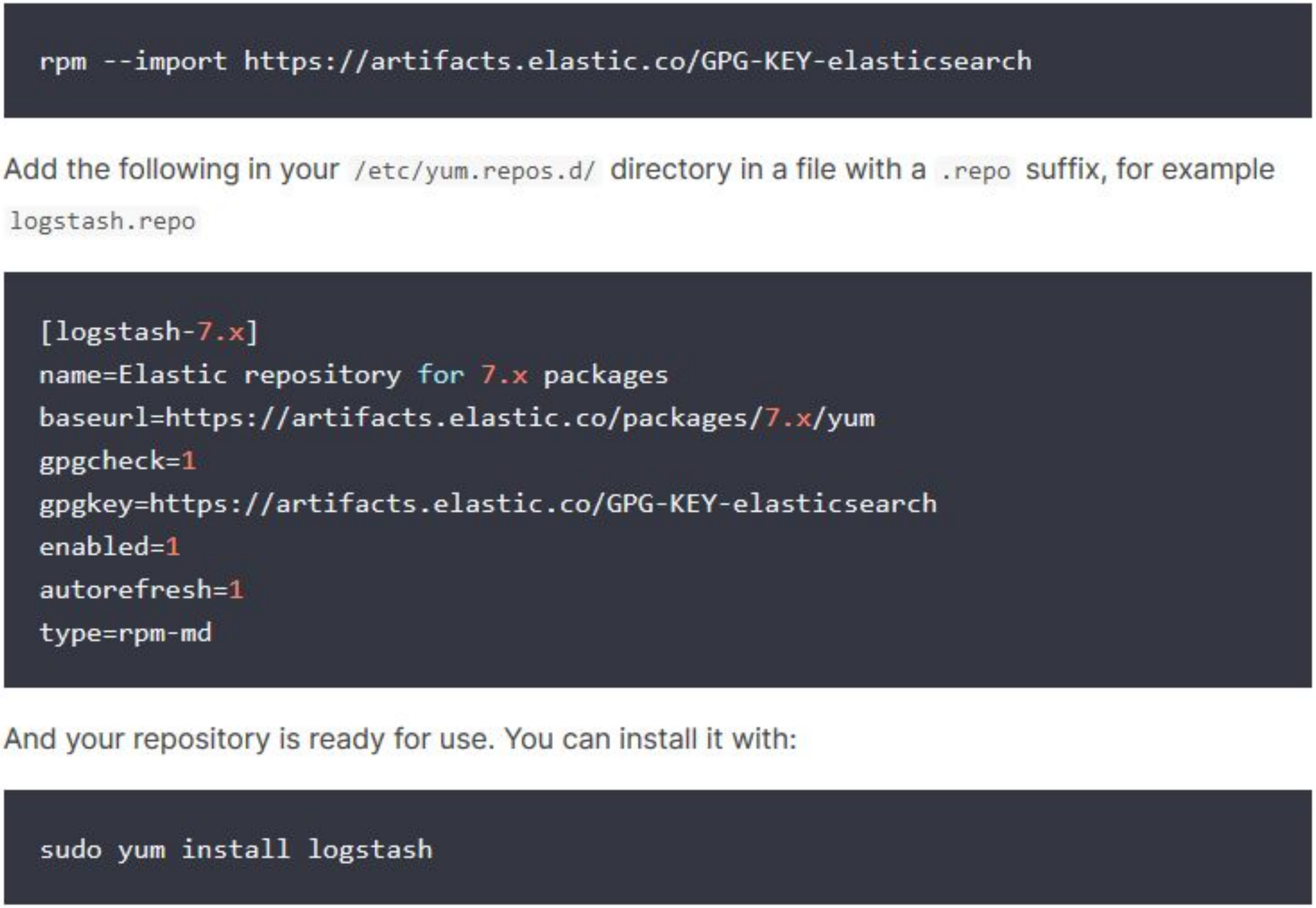
```
[root@elk-stack ~]# systemctl daemon-reload
[root@elk-stack ~]# systemctl start elasticsearch
[root@elk-stack ~]# systemctl enable elasticsearch
```

To check what are the indexes available in the ElasticSearch:

```
[root@elk-stack ~]# curl -XGET ‘http://localhost:9200/_cat/indices?v’

health status index          uuid                   pri rep docs.count docs.deleted store.size
pri.store.size
yellow open   stat-20180603 byH89tWFQSS_R9kS_QPGPw   5   1   54822544     0       6.9gb          6.9gb
yellow open   stat-20180616 qZYSua4CQDa18GGMc8uiHQ   5   1   51830338     0       6.6gb          6.6gb
yellow open   stat-20180604 PYdGUxX7SZ2aaFRV-ng4NQ   5   1   57828976     0       7.3gb          7.3gb
yellow open   stat-20180630 FwrBuf6FQ-6SlyZhknATLQ   5   1   50014372     0       6.4gb          6.4gb
yellow open   stat-20180618 _Nloca3jROCQ2vChWmDoGw   5   1   54976264     0         7gb            7gb
yellow open   stat-20180526 ObGvcFbfTDuuk_MtZNlCQA   5   1   51836183     0       6.6gb          6.6gb
yellow open   stat-20180615 t_CxQoauRUiVRTaJRPz2eQ   5   1   55490519     0         7gb            7gb
```

Logstash Installation

Logstash is one of the softwares inside the ELK stack. The main objective for this software is to convert NetFlow data into ElasticSearch acceptable format.



```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

Add the following in your `/etc/yum.repos.d/` directory in a file with a `.repo` suffix, for example `logstash.repo`

```
[logstash-7.x]
name=Elastic repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

And your repository is ready for use. You can install it with:

```
sudo yum install logstash
```

Configure Logstash to decode NetFlow

```
LS_HOME/bin/logstash-plugin install logstash-codec-sflow
LS_HOME/bin/logstash-plugin update logstash-codec-netflow
LS_HOME/bin/logstash-plugin update logstash-input-udp
LS_HOME/bin/logstash-plugin update logstash-filter-dns
```

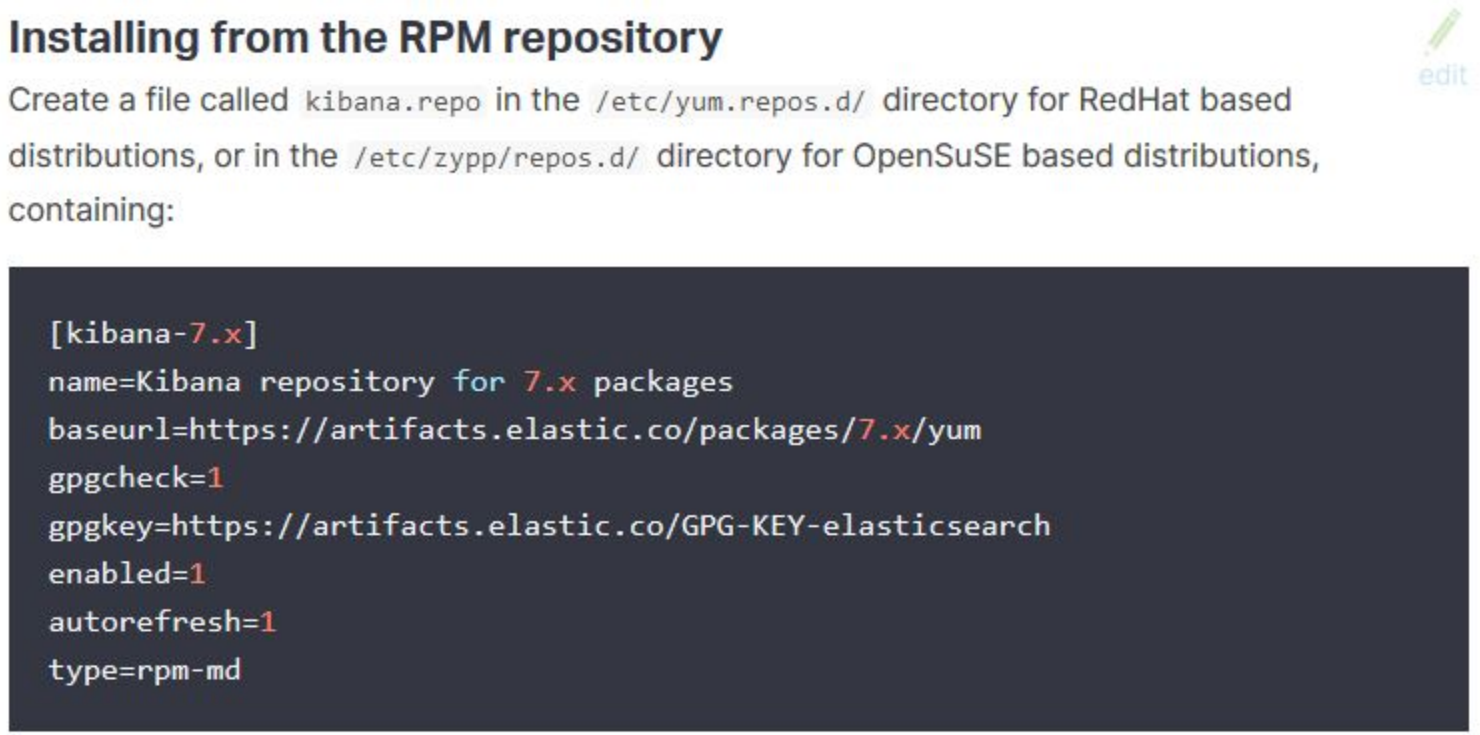
Create a netflow.conf /etc/logstash/

```
input {
  udp {
    port => 2055
    codec => netflow
  }
}

output {
  elasticsearch {
    protocol => "http"
    host => "127.0.0.1"
  }
  stdout { codec => rubydebug  }
}
```

Complete instruction:
https://www.elastic.co/guide/en/logstash/current/plugins-codecs-netflow.html

Kibana Installation

Kibana is one of the GUI tools that helps retrieve data from ElasticSearch. It can also come with the graphing capability to manipulate the Doc in ElasticSearch to be something more meaningful to system engineers.



Installing from the RPM repository

Create a file called `kibana.repo` in the `/etc/yum.repos.d/` directory for RedHat based distributions, or in the `/etc/zypp/repos.d/` directory for OpenSuSE based distributions, containing:

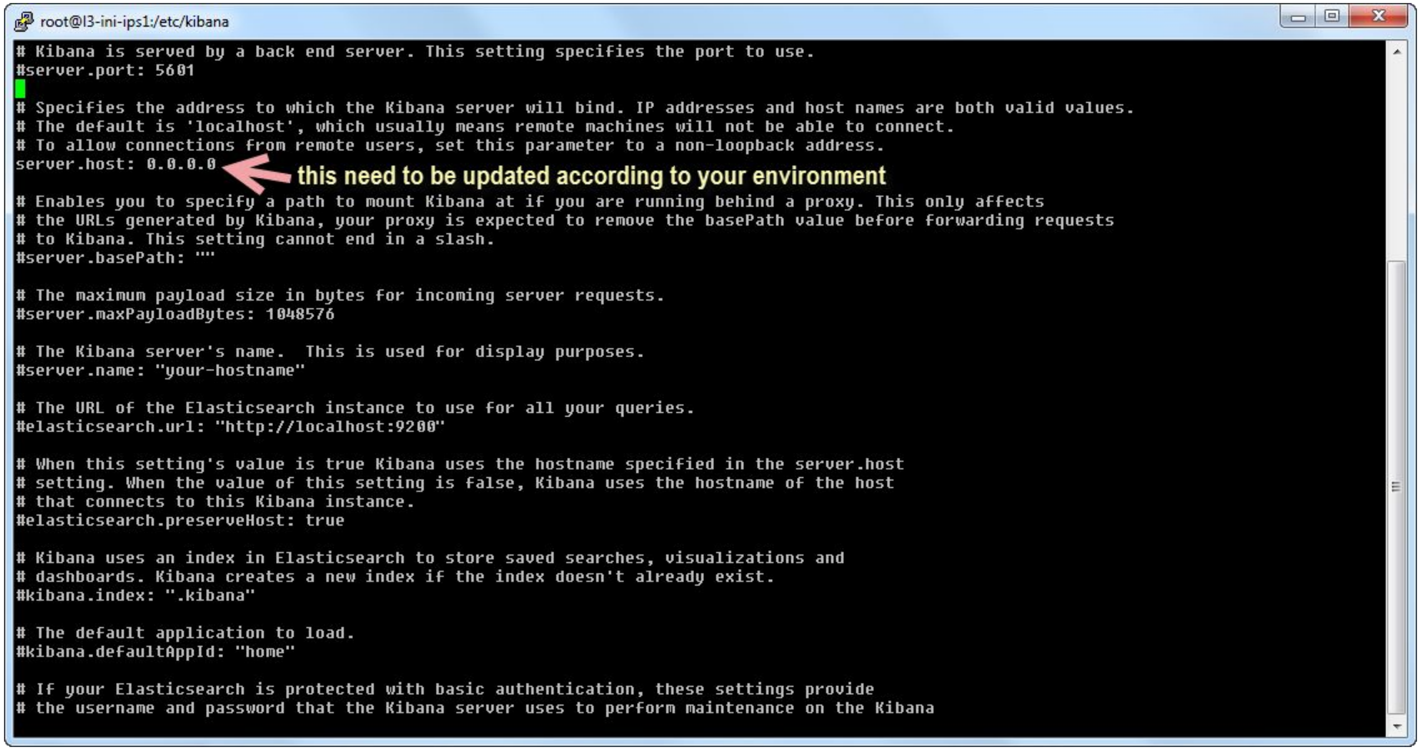
```
[kibana-7.x]
name=Kibana repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

```
sudo yum install kibana
```

Kibana Configuration

Kibana does not listen to any IP besides 127.0.0.1;
you will need to update the configuration file to make the Kibana accessible from outside the host.

```
vi /etc/kibana/kibana.yml
```

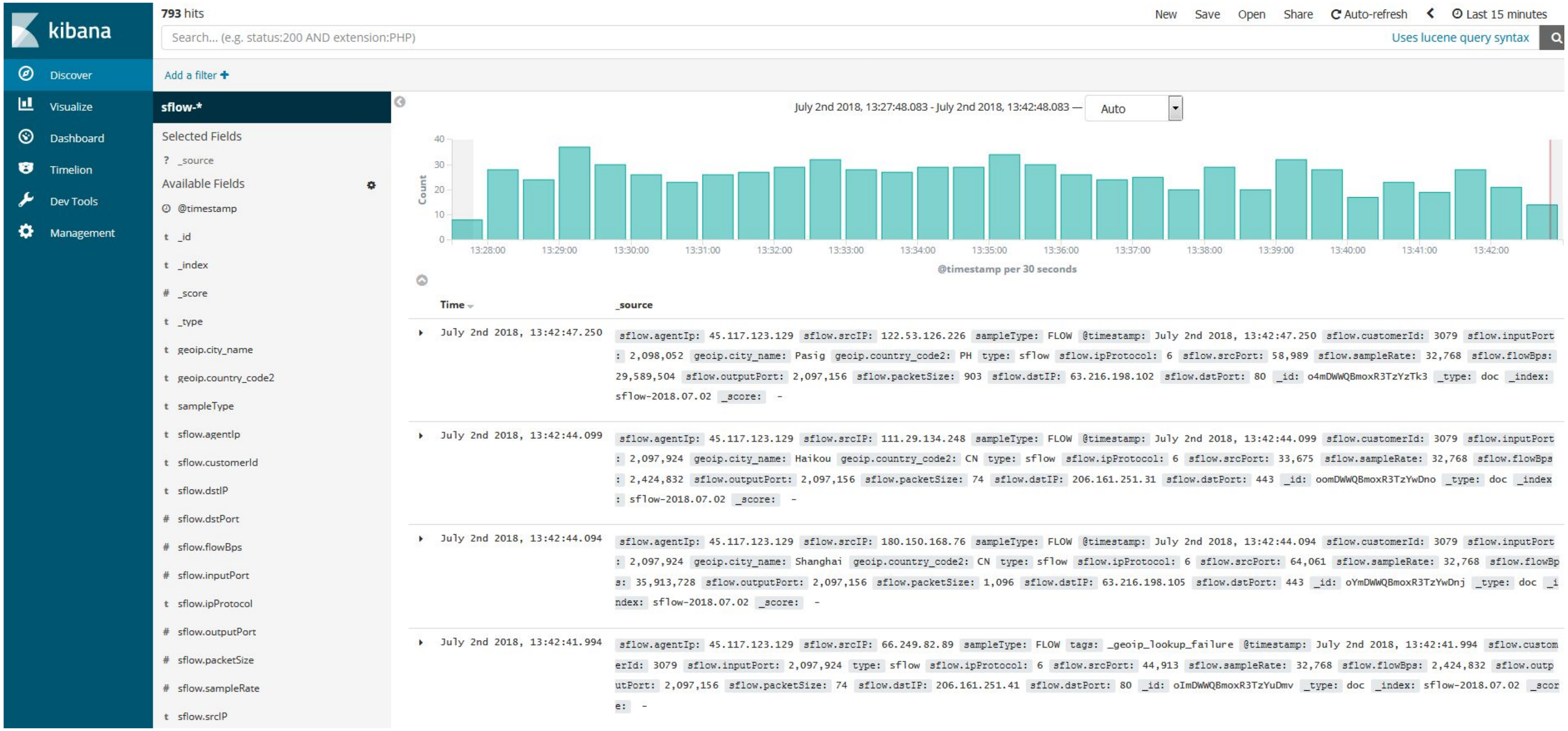


```
# Kibana is served by a back end server. This setting specifies the port to use.
#server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: 0.0.0.0

# Enables you to specify a path to mount Kibana at if you are running behind a proxy. This only affects
# the URLs generated by Kibana, your proxy is expected to remove the basePath value before forwarding requests
# to Kibana. This setting cannot end in a slash.
#server.basePath: ""

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name.  This is used for display purposes.
#server.name: "your-hostname"

# The URL of the Elasticsearch instance to use for all your queries.
#elasticsearch.url: "http://localhost:9200"

# When this setting's value is true Kibana uses the hostname specified in the server.host
# setting. When the value of this setting is false, Kibana uses the hostname of the host
# that connects to this Kibana instance.
#elasticsearch.preserveHost: true

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
#kibana.index: ".kibana"

# The default application to load.
#kibana.defaultAppId: "home"

# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
```

this need to be updated according to your environment

A quick look on the data stored in ElasticSearch

If the data **is successfully collected by Logstash**, this is what will be shown in Kibana:

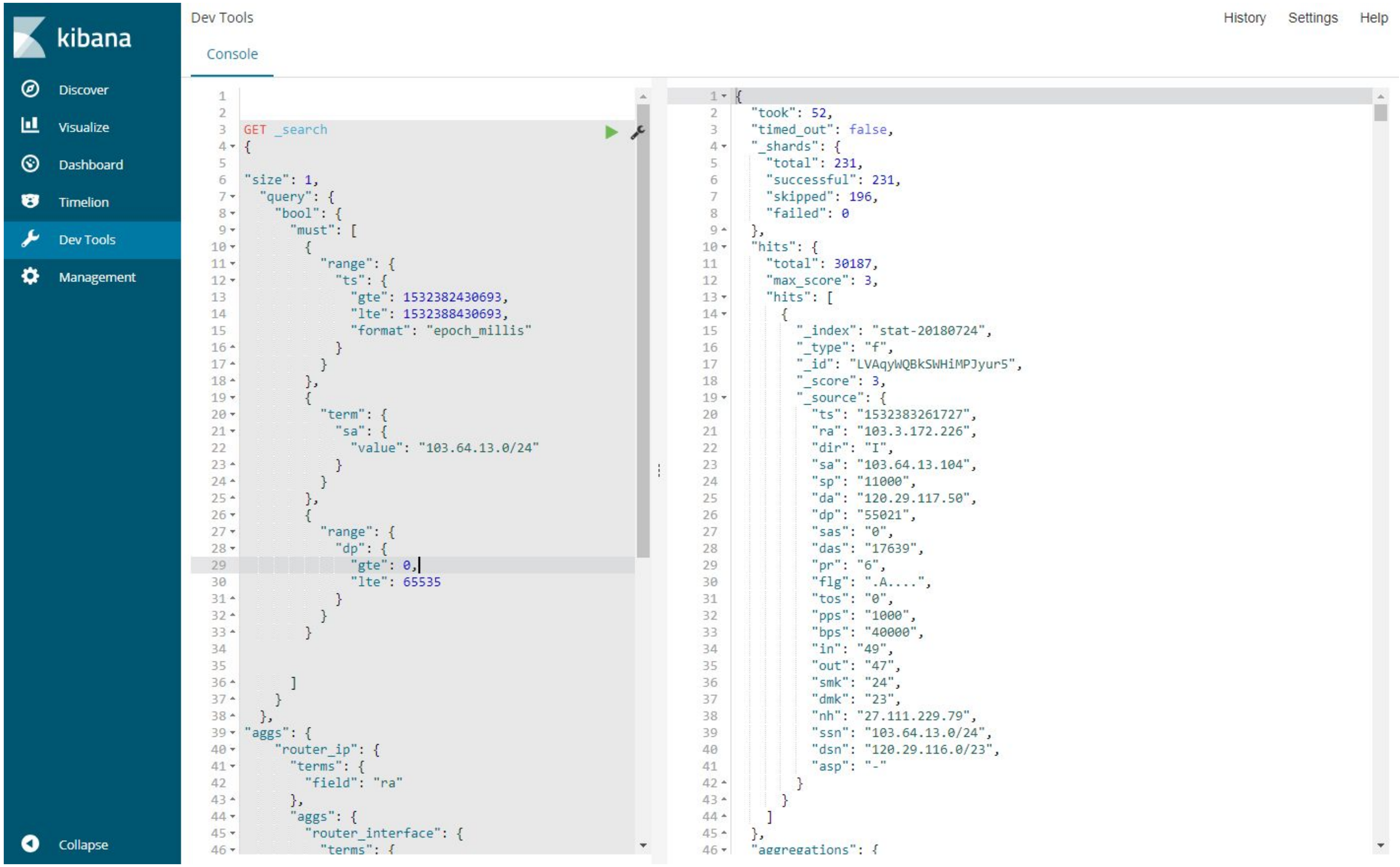


```
July 2nd 2018, 13:42:47.250  sflow.agentIp: 45.117.123.129 sflow.srcIP: 122.53.126.226 sampleType: FLOW @timestamp: July 2nd 2018, 13:42:47.250 sflow.customerId: 3079 sflow.inputPort
                             : 2,098,052 geoip.city_name: Pasig geoip.country_code2: PH type: sflow sflow.ipProtocol: 6 sflow.srcPort: 58,989 sflow.sampleRate: 32,768 sflow.flowBps:
                             29,589,504 sflow.outputPort: 2,097,156 sflow.packetSize: 903 sflow.dstIP: 63.216.198.102 sflow.dstPort: 80 _id: o4mDWWQBmoxR3TzYzTk3 _type: doc _index:
                             sflow-2018.07.02 _score: -

July 2nd 2018, 13:42:44.099  sflow.agentIp: 45.117.123.129 sflow.srcIP: 111.29.134.248 sampleType: FLOW @timestamp: July 2nd 2018, 13:42:44.099 sflow.customerId: 3079 sflow.inputPort
                             : 2,097,924 geoip.city_name: Haikou geoip.country_code2: CN type: sflow sflow.ipProtocol: 6 sflow.srcPort: 33,675 sflow.sampleRate: 32,768 sflow.flowBps
                             : 2,424,832 sflow.outputPort: 2,097,156 sflow.packetSize: 74 sflow.dstIP: 206.161.251.31 sflow.dstPort: 443 _id: oomDWWQBmoxR3TzYwDno _type: doc _index
                             : sflow-2018.07.02 _score: -

July 2nd 2018, 13:42:44.094  sflow.agentIp: 45.117.123.129 sflow.srcIP: 180.150.168.76 sampleType: FLOW @timestamp: July 2nd 2018, 13:42:44.094 sflow.customerId: 3079 sflow.inputPort
                             : 2,097,924 geoip.city_name: Shanghai geoip.country_code2: CN type: sflow sflow.ipProtocol: 6 sflow.srcPort: 64,061 sflow.sampleRate: 32,768 sflow.flowBp
                             s: 35,913,728 sflow.outputPort: 2,097,156 sflow.packetSize: 1,096 sflow.dstIP: 63.216.198.105 sflow.dstPort: 443 _id: oYmDWWQBmoxR3TzYwDnj _type: doc _i
                             ndex: sflow-2018.07.02 _score: -

July 2nd 2018, 13:42:41.994  sflow.agentIp: 45.117.123.129 sflow.srcIP: 66.249.82.89 sampleType: FLOW tags: _geoip_lookup_failure @timestamp: July 2nd 2018, 13:42:41.994 sflow.custom
                             erId: 3079 sflow.inputPort: 2,097,924 type: sflow sflow.ipProtocol: 6 sflow.srcPort: 44,913 sflow.sampleRate: 32,768 sflow.flowBps: 2,424,832 sflow.outp
                             utPort: 2,097,156 sflow.packetSize: 74 sflow.dstIP: 206.161.251.41 sflow.dstPort: 80 _id: oImDWWQBmoxR3TzYuDmv _type: doc _index: sflow-2018.07.02 _scor
                             e: -
```

How to query ElasticSearch **for top 10 IP talkers?**

ElasticSearch has it's own Query Language called Query DSL

Here is a sample query command for the IP range 103.64.13.0/24 at the specific time period. **(formatted in epoch milliseconds)**



```
GET _search
{

"size": 1,
  "query": {
    "bool": {
      "must": [
        {
          "range": {
            "ts": {
              "gte": 1532382430693,
              "lte": 1532388430693,
              "format": "epoch_millis"
            }
          }
        },
        {
          "term": {
            "sa": {
              "value": "103.64.13.0/24"
            }
          }
        },
        {
          "range": {
            "dp": {
              "gte": 0,
              "lte": 65535
            }
          }
        }


      ]
    }
  },
"aggs": {
    "router_ip": {
      "terms": {
        "field": "ra"
      },
      "aggs": {
        "router_interface": {
          "terms": {
```

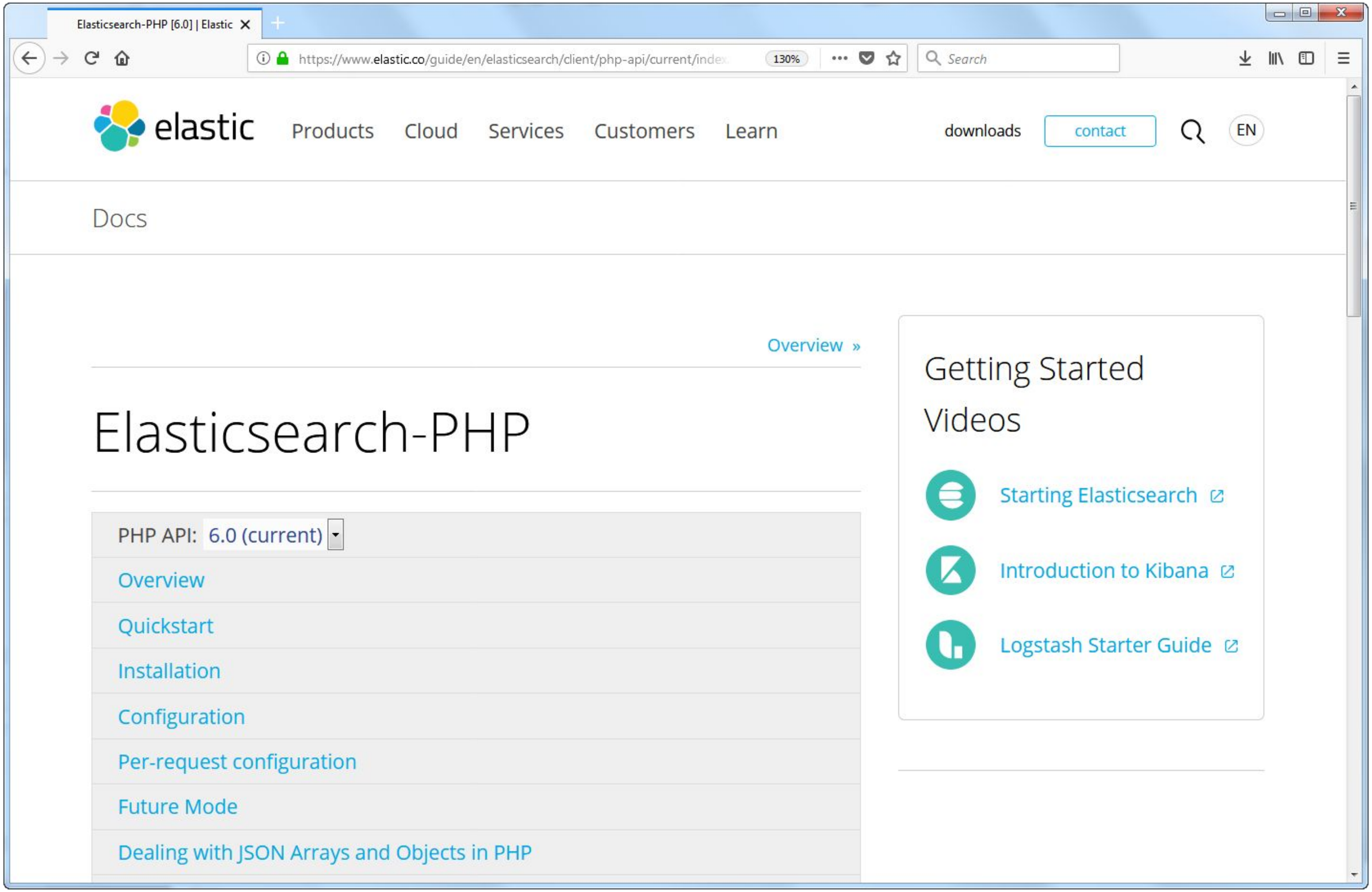
```
{
  "took": 52,
  "timed_out": false,
  "_shards": {
    "total": 231,
    "successful": 231,
    "skipped": 196,
    "failed": 0
  },
  "hits": {
    "total": 30187,
    "max_score": 3,
    "hits": [
      {
        "_index": "stat-20180724",
        "_type": "f",
        "_id": "LVAqyWQBkSWHiMPJyur5",
        "_score": 3,
        "_source": {
          "ts": "1532383261727",
          "ra": "103.3.172.226",
          "dir": "I",
          "sa": "103.64.13.104",
          "sp": "11000",
          "da": "120.29.117.50",
          "dp": "55021",
          "sas": "0",
          "das": "17639",
          "pr": "6",
          "flg": ".A....",
          "tos": "0",
          "pps": "1000",
          "bps": "40000",
          "in": "49",
          "out": "47",
          "smk": "24",
          "dmk": "23",
          "nh": "27.111.229.79",
          "ssn": "103.64.13.0/24",
          "dsn": "120.29.116.0/23",
          "asp": "-"
        }
      }
    ]
  },
  "aggregations": {
```

Kibana is easy to use…

However, it’s still complicated for my NOC team

We make use of ElasticSearch Client API for PHP, to make a query interface so that they can do the job quicker and simplify the learning curve.

A PHP client to consume ElasticSearch


Elasticsearch-PHP [6.0] | Elastic
https://www.elastic.co/guide/en/elasticsearch/client/php-api/current/index
130%
Search
elastic
Products
Cloud
Services
Customers
Learn
downloads
contact
EN
Docs
Overview »

Elasticsearch-PHP

PHP API: 6.0 (current)

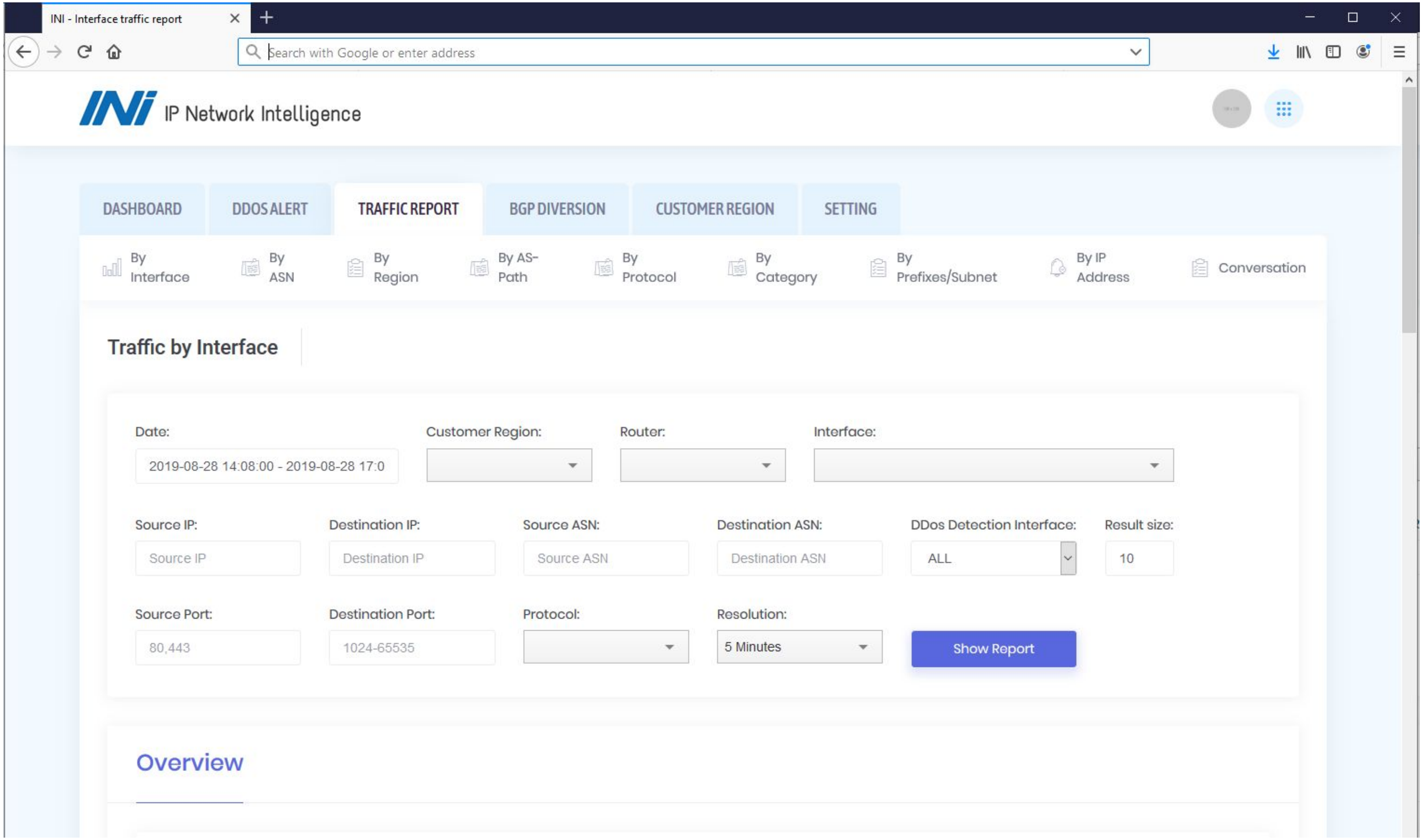
- Overview
- Quickstart
- Installation
- Configuration
- Per-request configuration
- Future Mode
- Dealing with JSON Arrays and Objects in PHP

Getting Started Videos

- Starting Elasticsearch
- Introduction to Kibana
- Logstash Starter Guide

IPDC
Your Trusted Cybersecurity Partner

A Query screen for the NOC engineer



Traffic by Interface

Date: 2019-08-28 14:08:00 - 2019-08-28 17:0

Customer Region:

Router:

Interface:

Source IP: Source IP

Destination IP: Destination IP

Source ASN: Source ASN

Destination ASN: Destination ASN

DDos Detection Interface: ALL

Result size: 10

Source Port: 80,443

Destination Port: 1024-65535

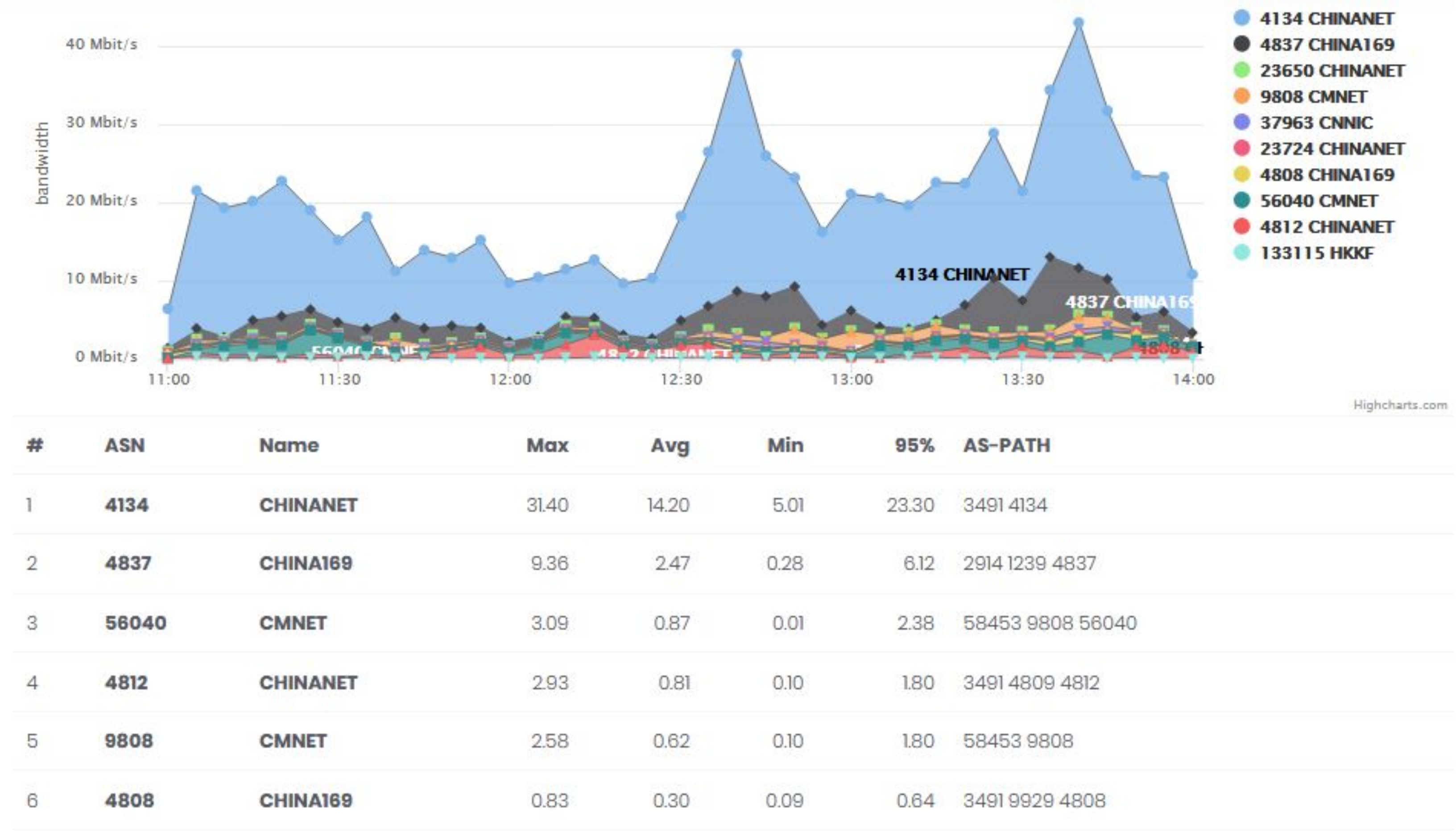
Protocol:

Resolution: 5 Minutes

Samples
on how we use the NetFlow Data

Outgoing traffic by ASN and it’s AS-PATH

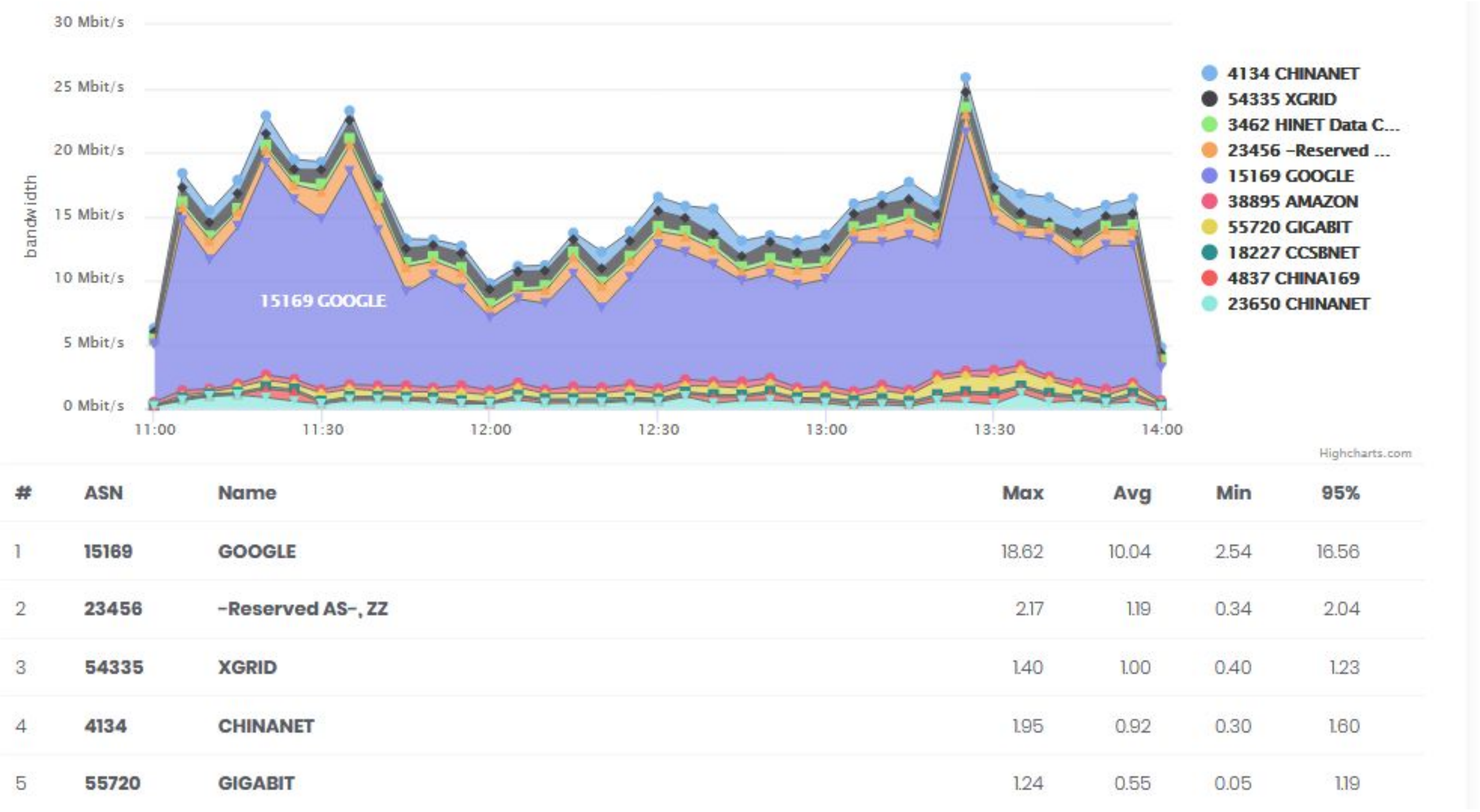
This allows us to know which ASN the traffic flows; and helps us optimize the planning and traffic engineering according to AS Number.



#	ASN	Name	Max	Avg	Min	95%	AS-PATH
1	**4134**	**CHINANET**	31.40	14.20	5.01	23.30	3491 4134
2	**4837**	**CHINA169**	9.36	2.47	0.28	6.12	2914 1239 4837
3	**56040**	**CMNET**	3.09	0.87	0.01	2.38	58453 9808 56040
4	**4812**	**CHINANET**	2.93	0.81	0.10	1.80	3491 4809 4812
5	**9808**	**CMNET**	2.58	0.62	0.10	1.80	58453 9808
6	**4808**	**CHINA169**	0.83	0.30	0.09	0.64	3491 9929 4808

Incoming traffic by Source ASN

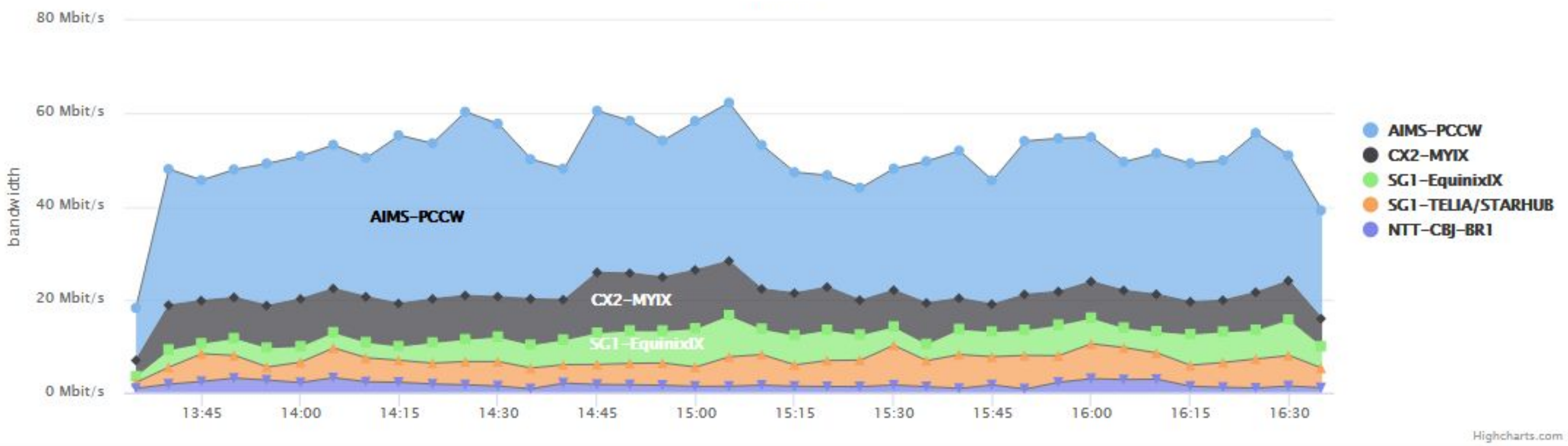
This is also helpful when it comes to traffic engineering



#	ASN	Name	Max	Avg	Min	95%
1	15169	GOOGLE	18.62	10.04	2.54	16.56
2	23456	-Reserved AS-, ZZ	2.17	1.19	0.34	2.04
3	54335	XGRID	1.40	1.00	0.40	1.23
4	4134	CHINANET	1.95	0.92	0.30	1.60
5	55720	GIGABIT	1.24	0.55	0.05	1.19

Identify customer traffic profile

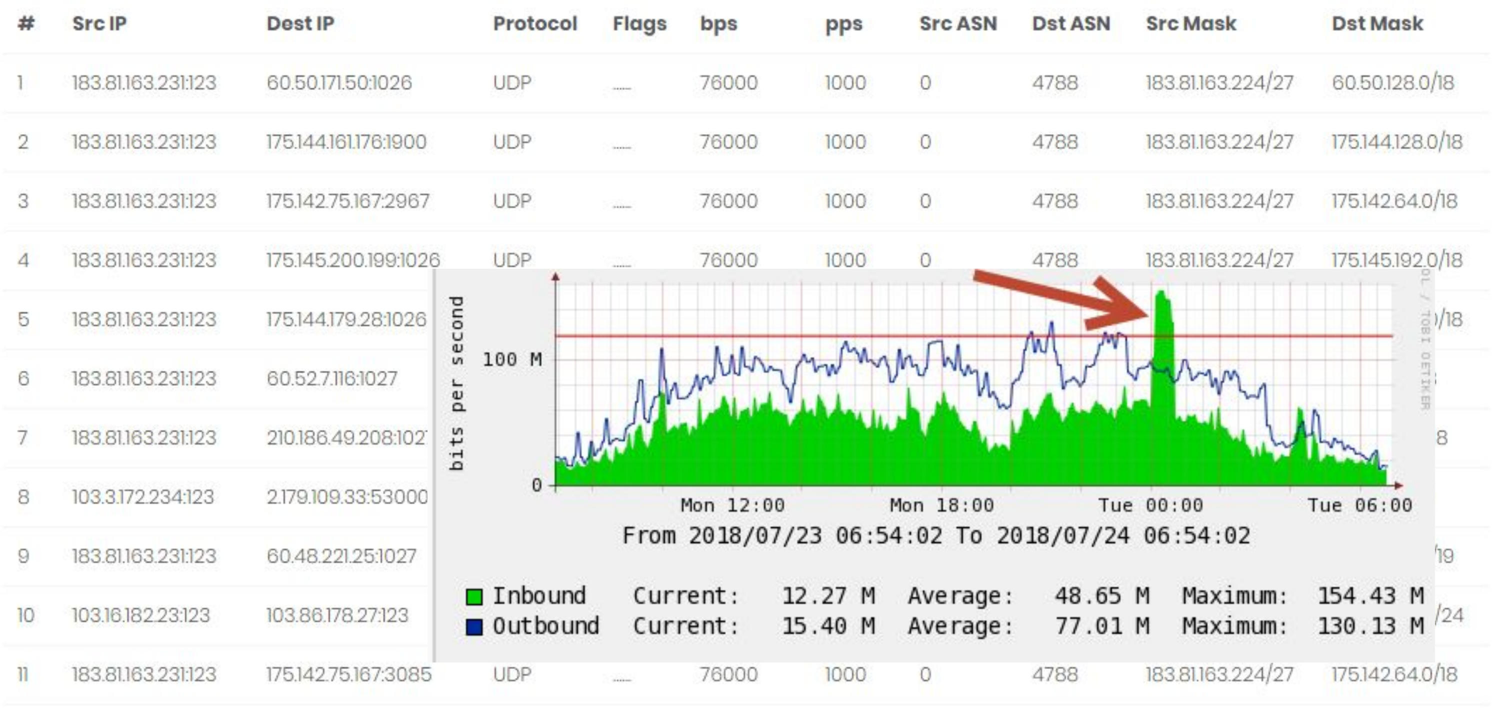
Identify the estimated bandwidth cost for each customer.
See if the customer traffic utilization is more towards international or local bandwidth.



#	Router IP	Name	Max	Avg	Min	95%
1	210.5.40.21	AIMS-PCCW	39.46	29.79	11.26	36.34
2	103.10.156.73	CX2-MYIX	12.93	8.70	3.38	12.35
3	103.3.172.227	SG1-TELIA/STARHUB	8.54	5.30	1.25	7.32
4	103.3.172.226	SG1-EquinixIX	8.94	5.22	1.38	7.84
5	103.21.181.2	NTT-CBJ-BR1	3.25	1.81	0.83	3.07

IP Conversation History

It’s something really useful for troubleshooting a network related issue, such as **spamming activity**, **NTP attack** within the network, and ability to **identify the compromised host** quickly.

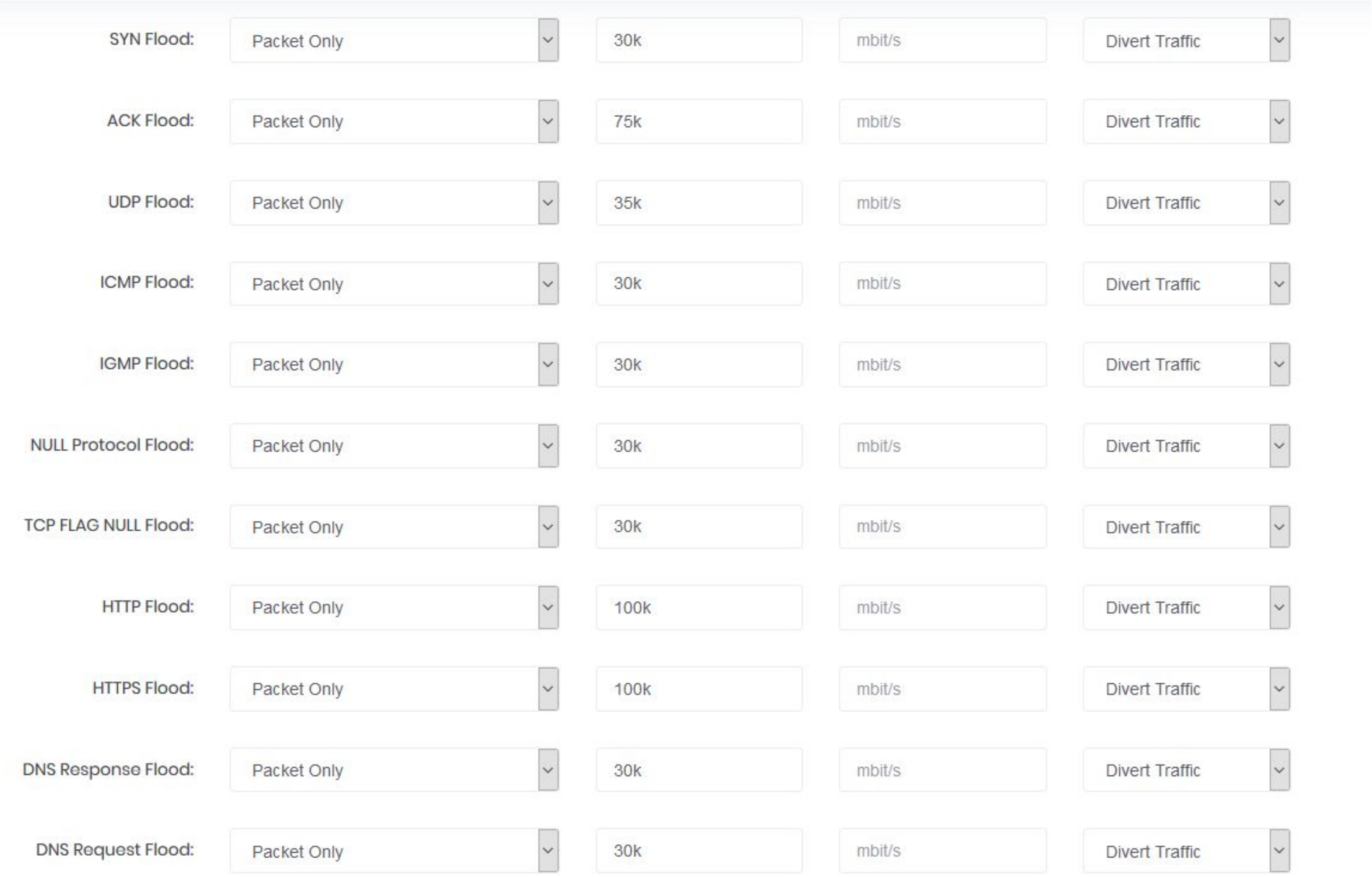


#	Src IP	Dest IP	Protocol	Flags	bps	pps	Src ASN	Dst ASN	Src Mask	Dst Mask
1	183.81.163.231:123	60.50.171.50:1026	UDP	......	76000	1000	0	4788	183.81.163.224/27	60.50.128.0/18
2	183.81.163.231:123	175.144.161.176:1900	UDP	......	76000	1000	0	4788	183.81.163.224/27	175.144.128.0/18
3	183.81.163.231:123	175.142.75.167:2967	UDP	......	76000	1000	0	4788	183.81.163.224/27	175.142.64.0/18
4	183.81.163.231:123	175.145.200.199:1026	UDP	......	76000	1000	0	4788	183.81.163.224/27	175.145.192.0/18
5	183.81.163.231:123	175.144.179.28:1026								)/18
6	183.81.163.231:123	60.52.7.116:1027								
7	183.81.163.231:123	210.186.49.208:102								8
8	103.3.172.234:123	2.179.109.33:53000								
9	183.81.163.231:123	60.48.221.25:1027								/19
10	103.16.182.23:123	103.86.178.27:123								/24
11	183.81.163.231:123	175.142.75.167:3085	UDP	......	76000	1000	0	4788	183.81.163.224/27	175.142.64.0/18

bits per second
100 M
0
Mon 12:00
Mon 18:00
Tue 00:00
Tue 06:00
From 2018/07/23 06:54:02 To 2018/07/24 06:54:02
🟩 Inbound Current: 12.27 M Average: 48.65 M Maximum: 154.43 M
🟦 Outbound Current: 15.40 M Average: 77.01 M Maximum: 130.13 M
OL / TOBI OETIKER

We also use the Netflow information to do DDOS-Detection & Mitigation

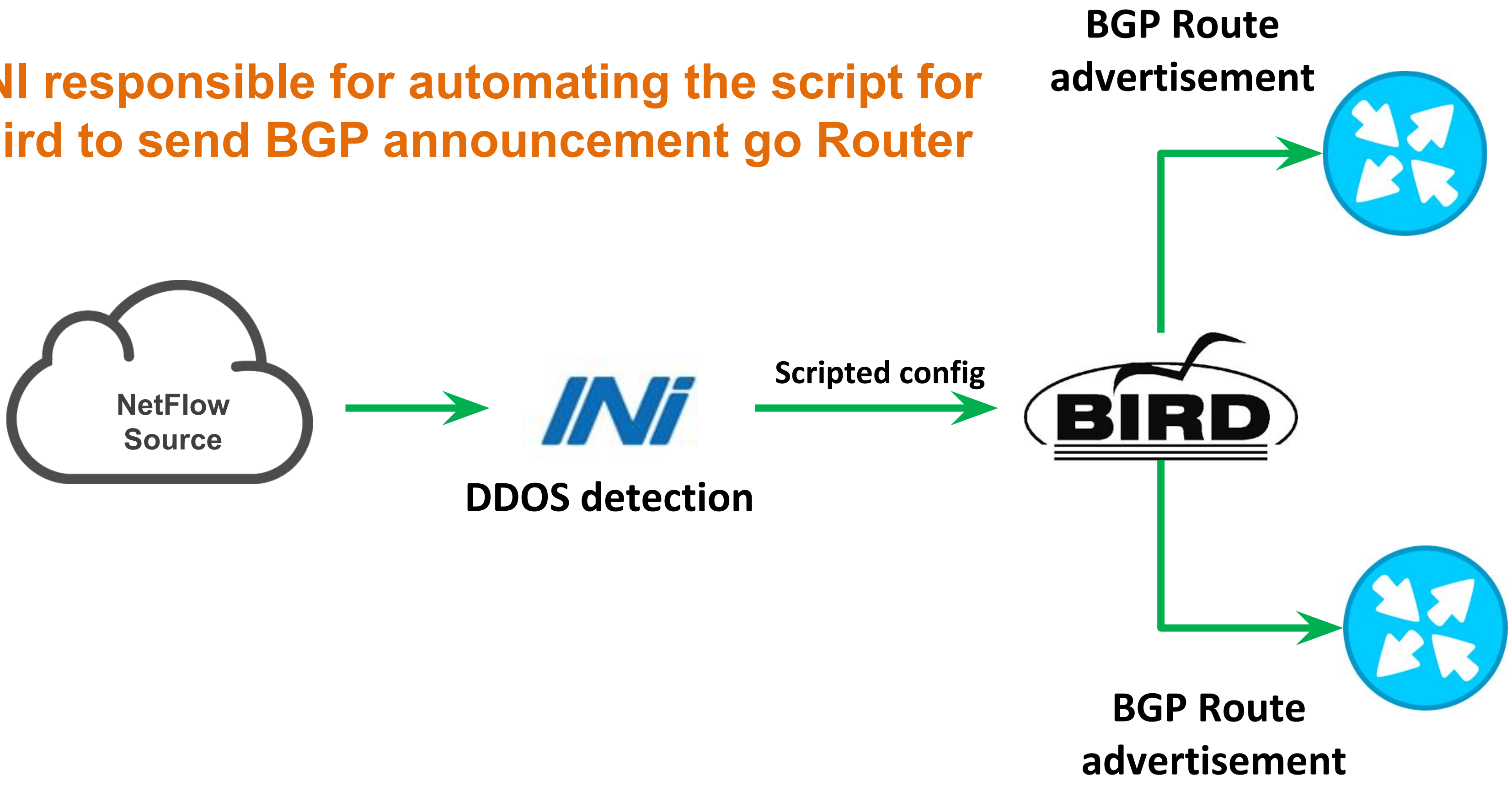
We wrote a utility (named INI) that analysize the netflow record; and when the threshold met, then the INI will trigger a BGP diversion



SYN Flood:	Packet Only	30k	mbit/s	Divert Traffic
ACK Flood:	Packet Only	75k	mbit/s	Divert Traffic
UDP Flood:	Packet Only	35k	mbit/s	Divert Traffic
ICMP Flood:	Packet Only	30k	mbit/s	Divert Traffic
IGMP Flood:	Packet Only	30k	mbit/s	Divert Traffic
NULL Protocol Flood:	Packet Only	30k	mbit/s	Divert Traffic
TCP FLAG NULL Flood:	Packet Only	30k	mbit/s	Divert Traffic
HTTP Flood:	Packet Only	100k	mbit/s	Divert Traffic
HTTPS Flood:	Packet Only	100k	mbit/s	Divert Traffic
DNS Response Flood:	Packet Only	30k	mbit/s	Divert Traffic
DNS Request Flood:	Packet Only	30k	mbit/s	Divert Traffic

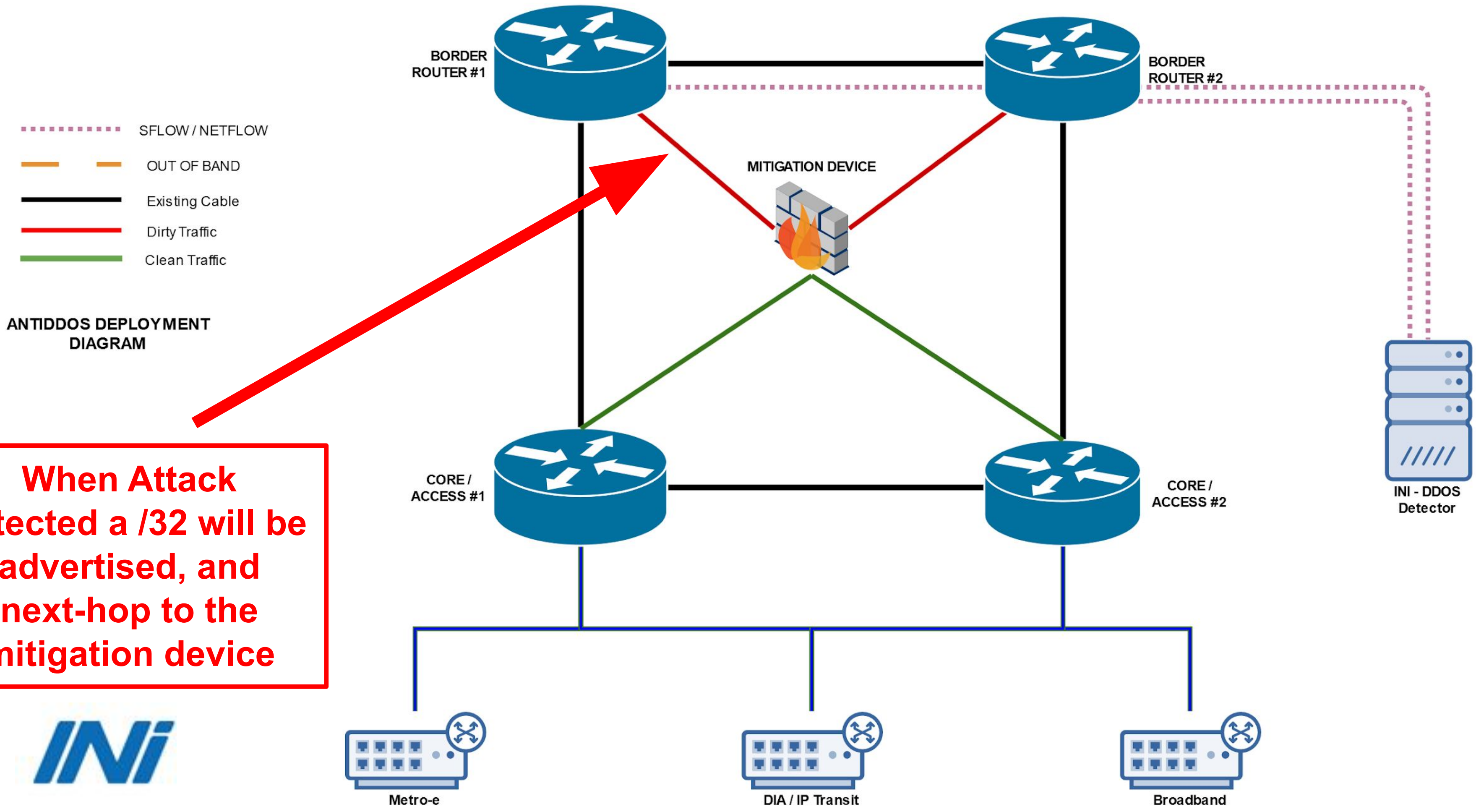
To mitigate the attack toward the victim IP.
We use BIRD routing daemon to communicate with our Borders routers.

INI responsible for automating the script for Bird to send BGP announcement go Router



Mitigation Method #1,
Clean the DDOS attack locally on prem device

I P D C
Your Trusted Cybersecurity Partner



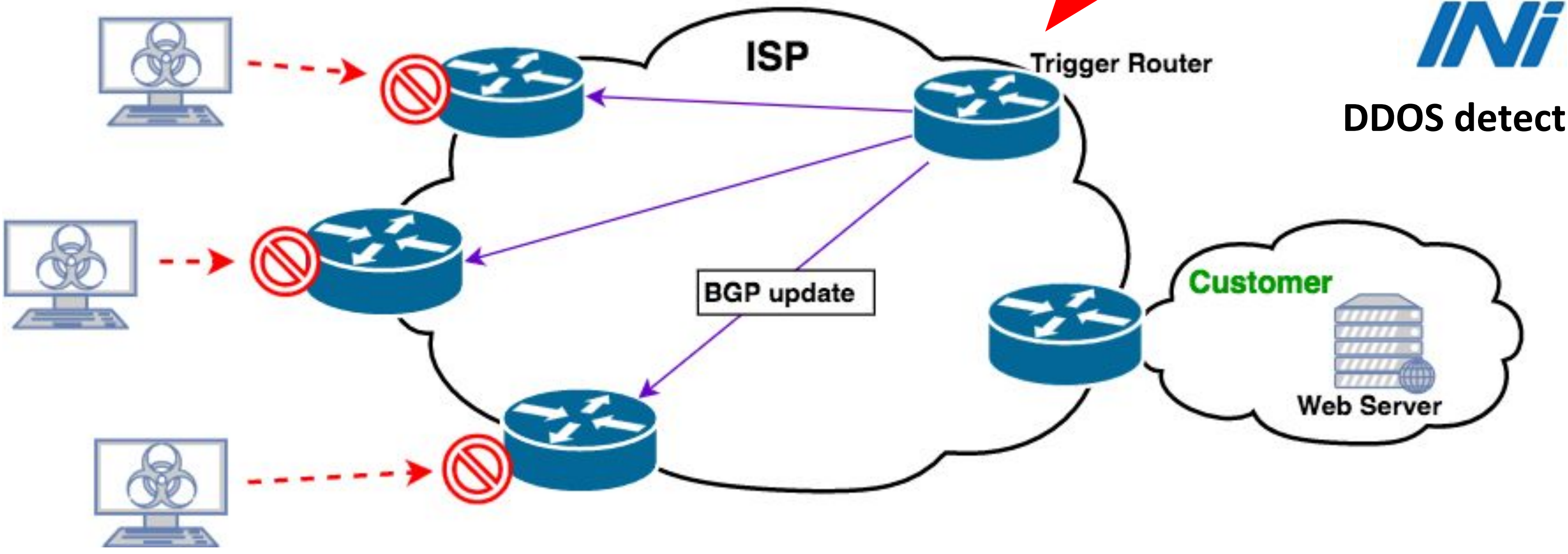
When Attack detected a /32 will be advertised, and next-hop to the mitigation device

INi

DDOS detection

Mitigation Method #2,
Send a Remote Triggered Blackhole command

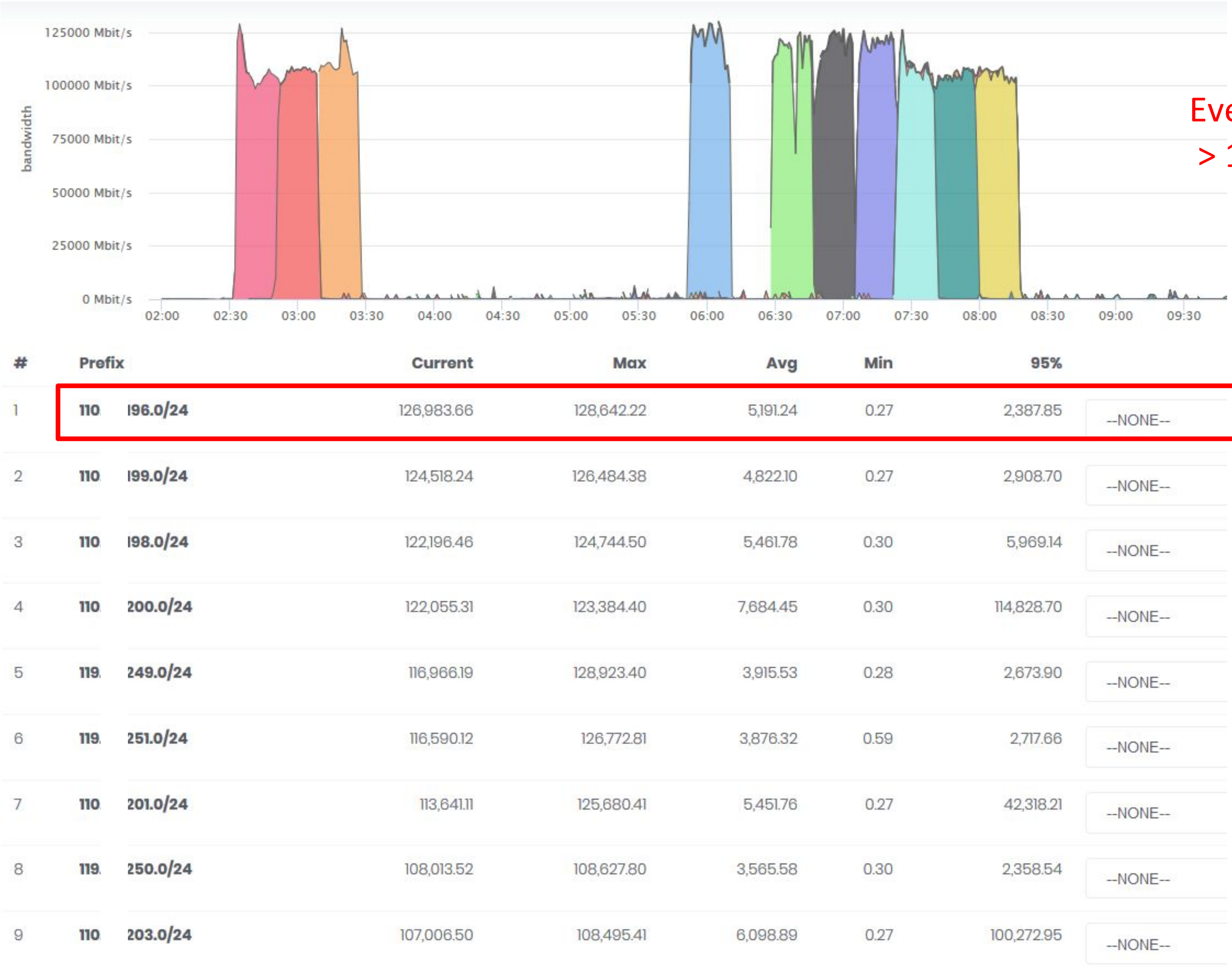
Or, it could trigger a remote blackhole
To the upstream provider



DDOS detection

2 of the trending pattern that we encountered.

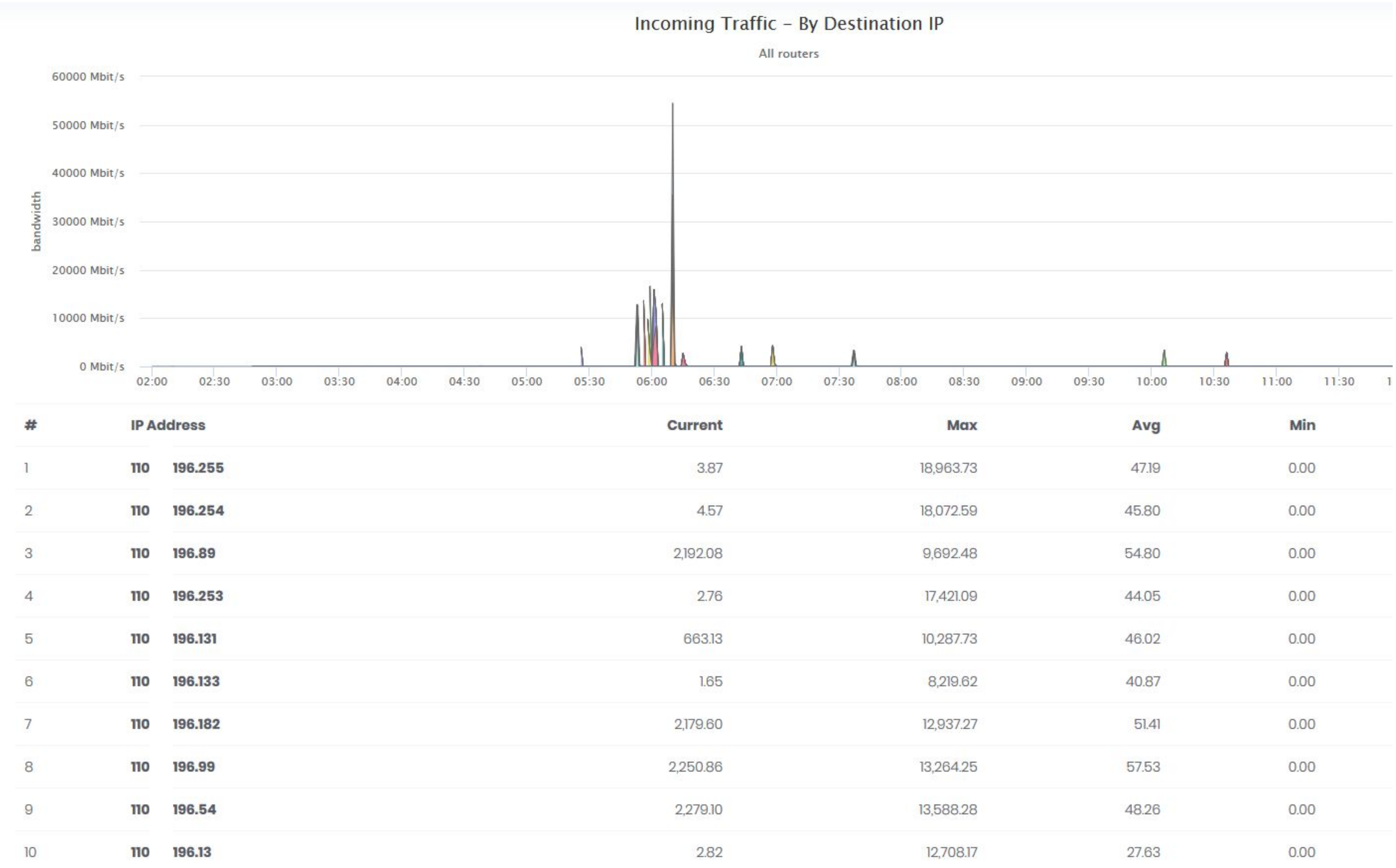
Trend #1: The attack is hitting all IPs in the subnet..



#	Prefix	Current	Max	Avg	Min	95%	
1	110 196.0/24	126,983.66	128,642.22	5,191.24	0.27	2,387.85	--NONE--
2	110 199.0/24	124,518.24	126,484.38	4,822.10	0.27	2,908.70	--NONE--
3	110 198.0/24	122,196.46	124,744.50	5,461.78	0.30	5,969.14	--NONE--
4	110 200.0/24	122,055.31	123,384.40	7,684.45	0.30	114,828.70	--NONE--
5	119. 249.0/24	116,966.19	128,923.40	3,915.53	0.28	2,673.90	--NONE--
6	119. 251.0/24	116,590.12	126,772.81	3,876.32	0.59	2,717.66	--NONE--
7	110 201.0/24	113,641.11	125,680.41	5,451.76	0.27	42,318.21	--NONE--
8	119. 250.0/24	108,013.52	108,627.80	3,565.58	0.30	2,358.54	--NONE--
9	110 203.0/24	107,006.50	108,495.41	6,098.89	0.27	100,272.95	--NONE--

Every /24 is experience
> 120Gbit/s attack

If we breakdown the usage by IP address by this subnet. We could see which IP is being hit between 8G – 19G



#	IP Address	Current	Max	Avg	Min
1	110 196.255	3.87	18,963.73	47.19	0.00
2	110 196.254	4.57	18,072.59	45.80	0.00
3	110 196.89	2,192.08	9,692.48	54.80	0.00
4	110 196.253	2.76	17,421.09	44.05	0.00
5	110 196.131	663.13	10,287.73	46.02	0.00
6	110 196.133	1.65	8,219.62	40.87	0.00
7	110 196.182	2,179.60	12,937.27	51.41	0.00
8	110 196.99	2,250.86	13,264.25	57.53	0.00
9	110 196.54	2,279.10	13,588.28	48.26	0.00
10	110 196.13	2.82	12,708.17	27.63	0.00

Trend #2: Carpet style attack

This attack method is crafted to send attack “below” the legitimate volume

Example:

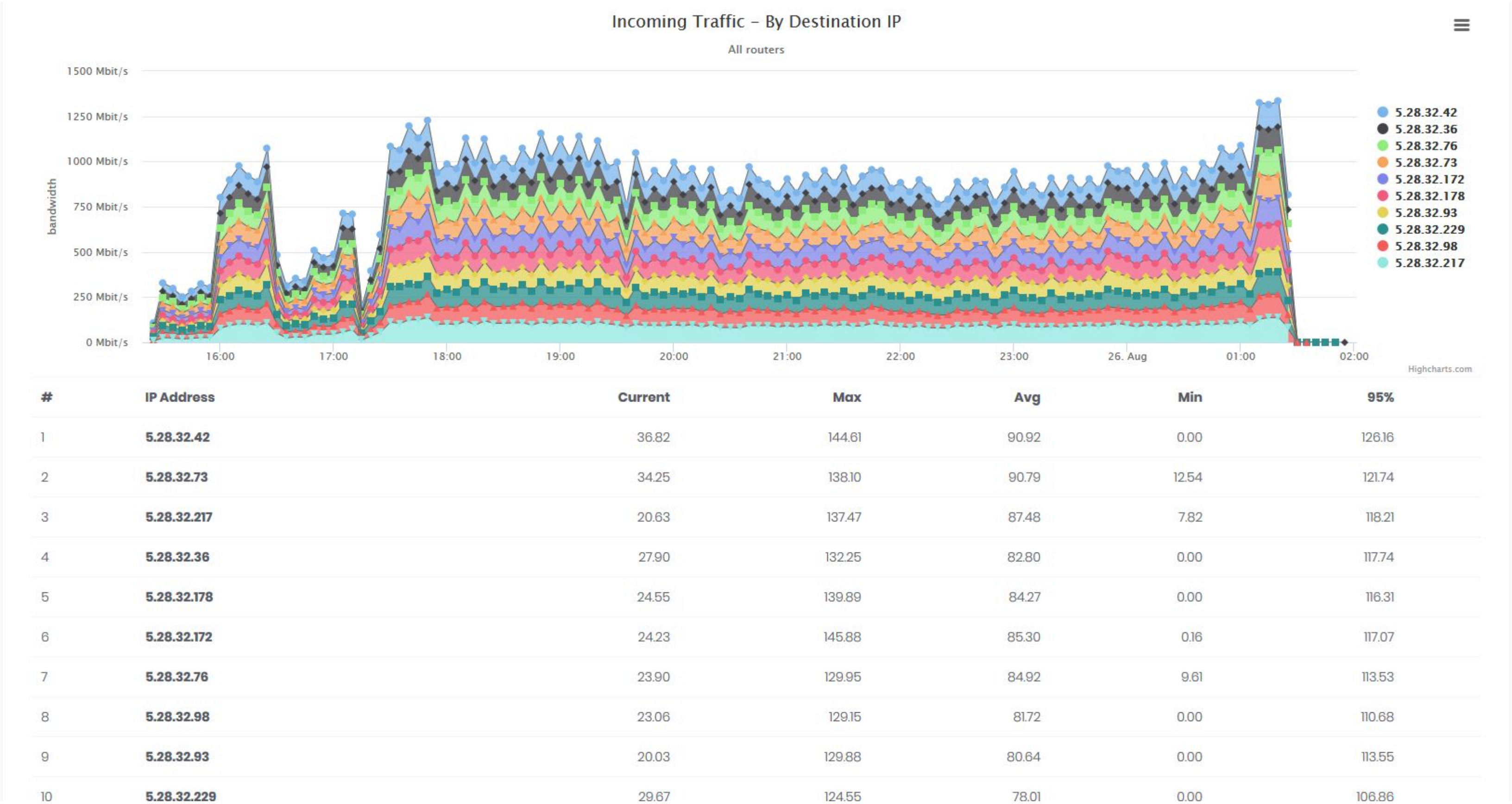
If you allocate 1 fixed IP with 50Mbit/s for each customer

How the attack being done is.
They will attack

45Mbit/s to each of your IP address. The total attack traffic would be

256 IP x 45Mbit/s =11.5Gbi/s
In some cases, they will spread the attack over a /22

Carpet style attack

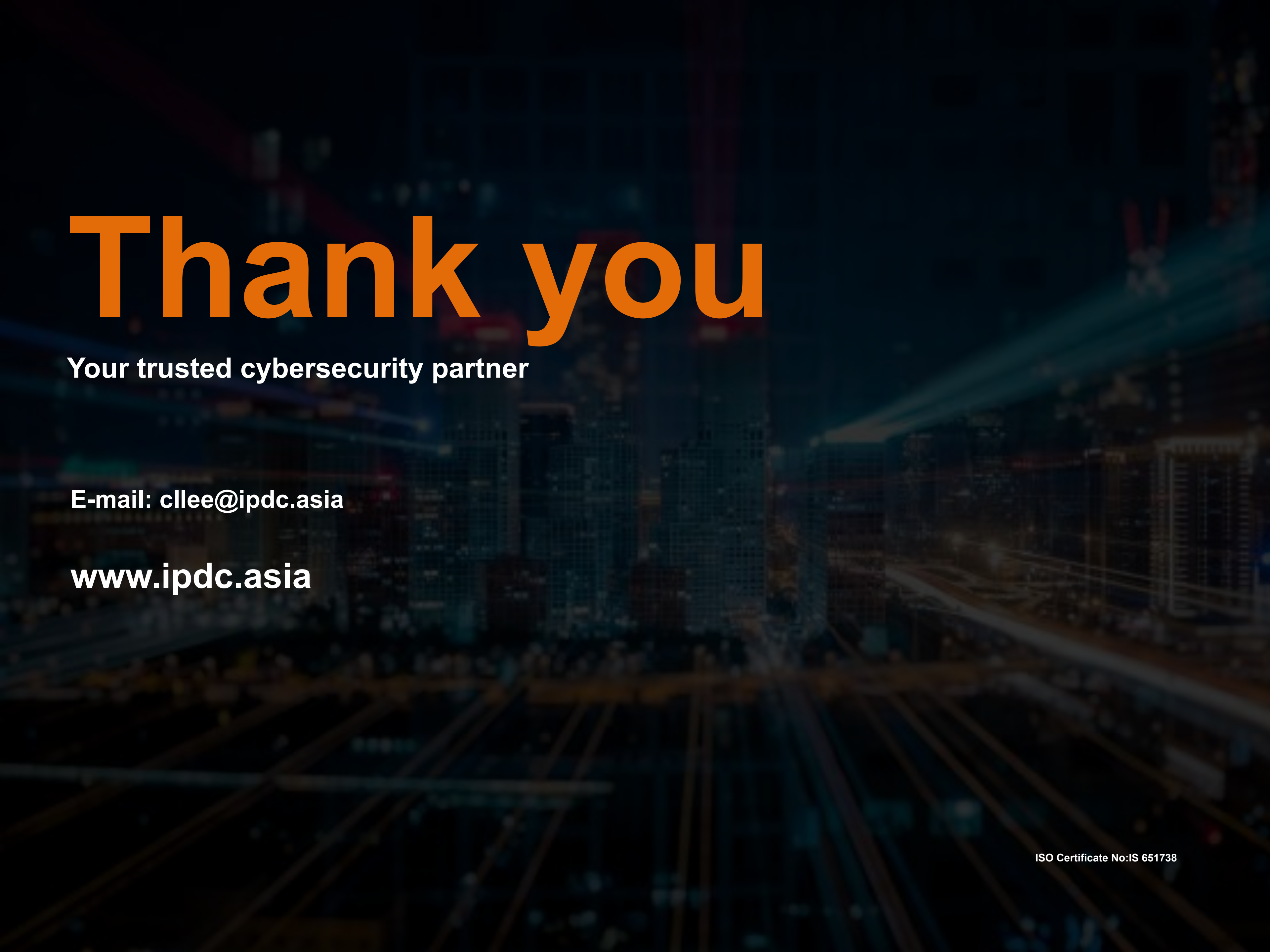


#	IP Address	Current	Max	Avg	Min	95%
1	**5.28.32.42**	36.82	144.61	90.92	0.00	126.16
2	**5.28.32.73**	34.25	138.10	90.79	12.54	121.74
3	**5.28.32.217**	20.63	137.47	87.48	7.82	118.21
4	**5.28.32.36**	27.90	132.25	82.80	0.00	117.74
5	**5.28.32.178**	24.55	139.89	84.27	0.00	116.31
6	**5.28.32.172**	24.23	145.88	85.30	0.16	117.07
7	**5.28.32.76**	23.90	129.95	84.92	9.61	113.53
8	**5.28.32.98**	23.06	129.15	81.72	0.00	110.68
9	**5.28.32.93**	20.03	129.88	80.64	0.00	113.55
10	**5.28.32.229**	29.67	124.55	78.01	0.00	106.86

Summary

1) Netflow would be very useful for traffic engineering & Analysis.
2) Storing them into ELK stack for graph plotting Is not difficult, and it's free with opensource tool

ANY QUESTIONS?



Thank you

Your trusted cybersecurity partner

E-mail: cllee@ipdc.asia

www.ipdc.asia

ISO Certificate No:IS 651738