PANTHEON
.tech

Network Flow Monitoring

in K8s with Contiv-VPP CNI and Elastic Stack

Rastislav Szabó
rastislav.szabo@pantheon.tech

About the Speaker – Rastislav Szabó

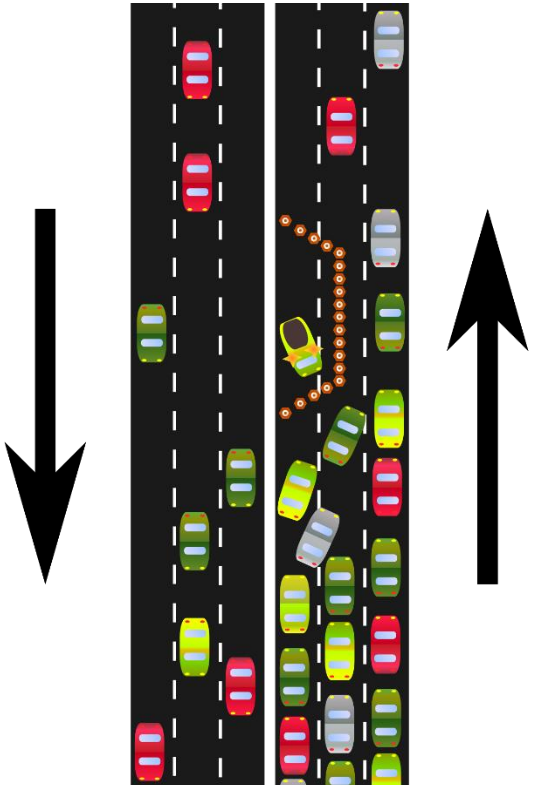
- Staff Engineer at PANTHEON.tech
- Architecting software solutions for networking industry

- Previously focused on network manageablity using NETCONF + YANG
- Currently working on cloud-native networking infrastructure projects

- Open-source contributions: Sysrepo, FD.io, Ligato, Contiv-VPP

Motivation for Network Monitoring in K8s

- Network failure identification & alerting
  - unexpected congestion, packet drop, ...
  - between pods on the same node & in the underlying network
- Identification of the bottlenecks
  - equal traffic distribution in the cluster
  - limits of large scale deployments
- Malicious activity detection & investigation
- CNFs (Cloud-Native Network Functions) deployments
  - all of the above becomes even more important

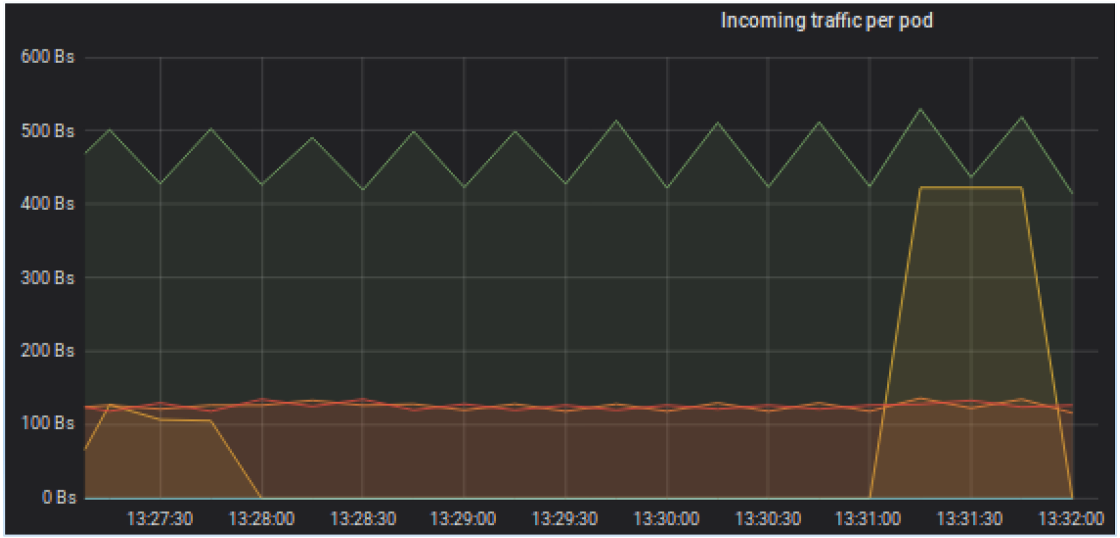


https://en.wikipedia.org/wiki/Traffic_bottleneck

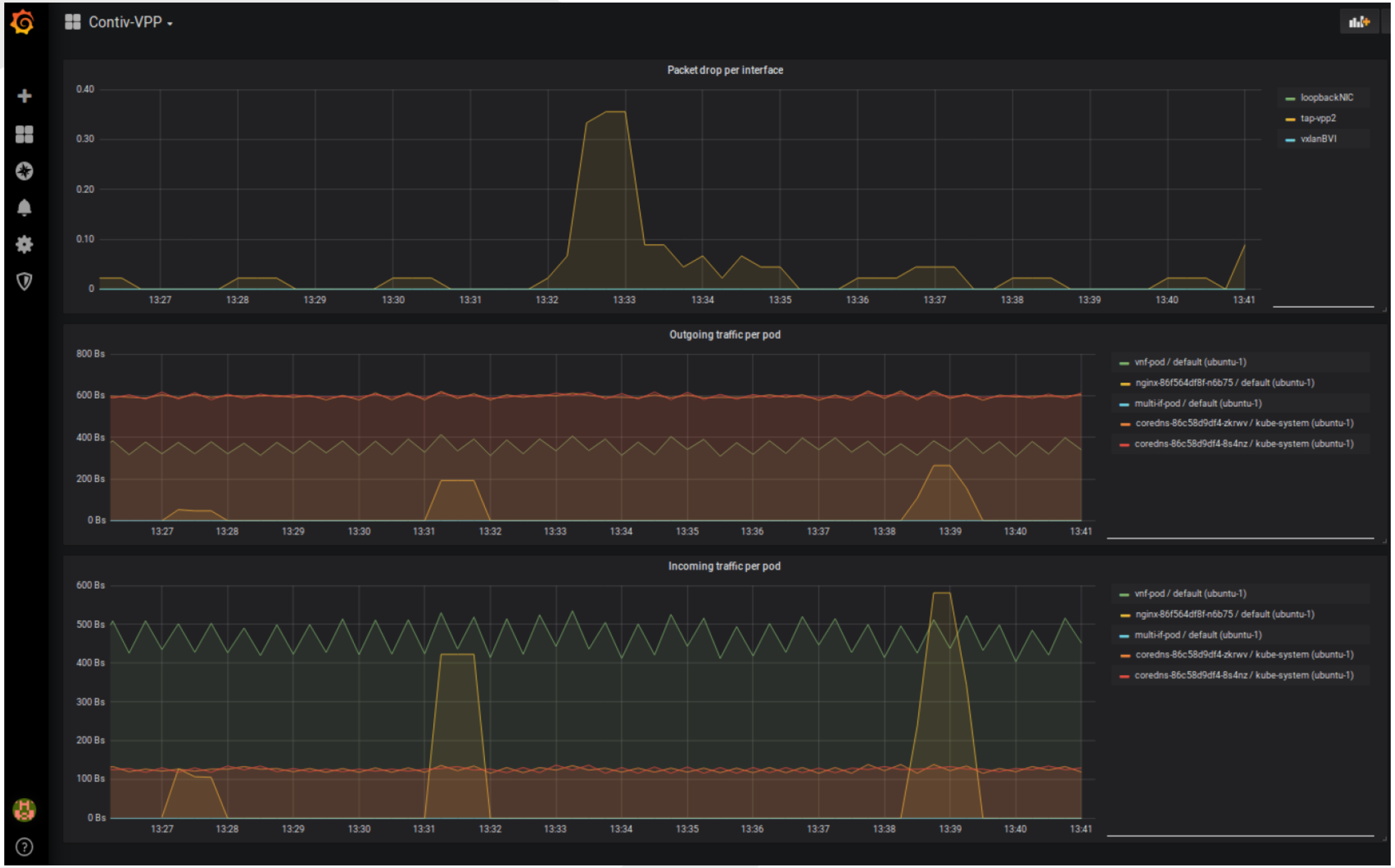
Options for Network Monitoring in K8s

- Metrics served by CNI plugins
  - many CNIs export metrics in Prometheus format
  - only some of them actually export helpful data
- Service mesh metrics
  - Istio can collect TCP telemetry data and export them via Prometheus
- DIY / 3rd party tool metrics
  - e.g. monitoring network interfaces within each pod's network namespace
- Metrics are not enough
  - not enough for deeper analysis, e.g. in case of security incidents

Prometheus + Grafana



Per-Pod Interface Metrics by Contiv-VPP CNI

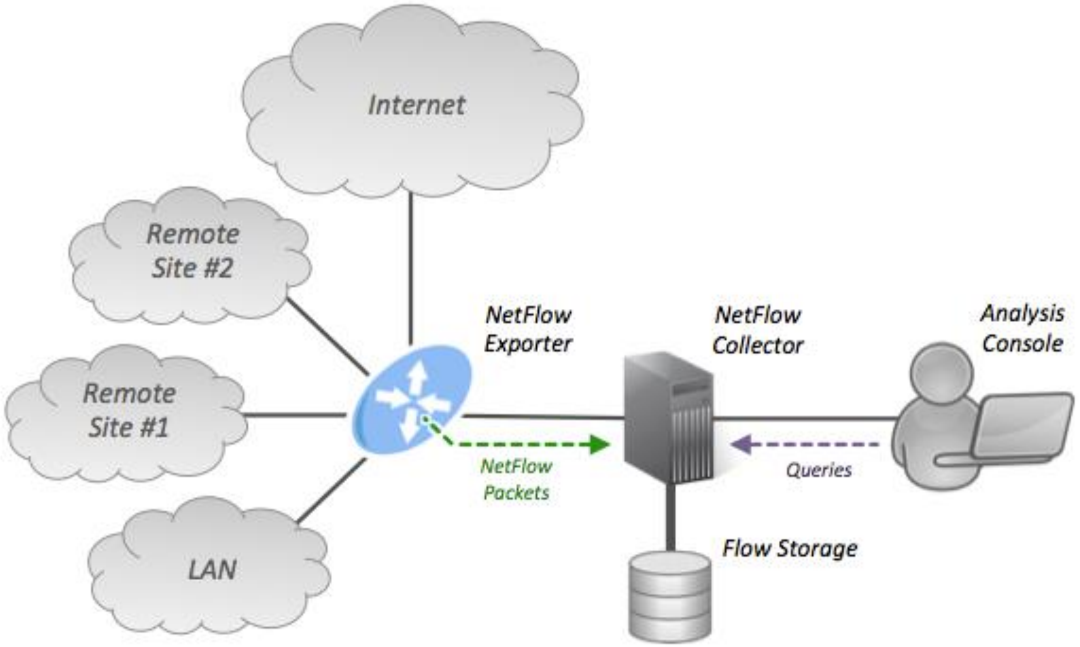

Contiv-VPP
Packet drop per interface
0.40
0.30
0.20
0.10
0
13:27
13:28
13:29
13:30
13:31
13:32
13:33
13:34
13:35
13:36
13:37
13:38
13:39
13:40
13:41
loopbackNIC
tap-vpp2
vxlanBVI
Outgoing traffic per pod
800 Bs
600 Bs
400 Bs
200 Bs
0 Bs
vnf-pod / default (ubuntu-1)
nginx-86f564df8f-n6b75 / default (ubuntu-1)
multi-if-pod / default (ubuntu-1)
coredns-86c58d9df4-zkrwv / kube-system (ubuntu-1)
coredns-86c58d9df4-8s4nz / kube-system (ubuntu-1)
Incoming traffic per pod
600 Bs
500 Bs
400 Bs
300 Bs
200 Bs
100 Bs
0 Bs
vnf-pod / default (ubuntu-1)
nginx-86f564df8f-n6b75 / default (ubuntu-1)
multi-if-pod / default (ubuntu-1)
coredns-86c58d9df4-zkrwv / kube-system (ubuntu-1)
coredns-86c58d9df4-8s4nz / kube-system (ubuntu-1)

- Good for generating alerts, spotting issues, etc.
- Cannot go back in the history and look e.g. at the details of the traffic that caused a spike on the graph

Network Monitoring in Traditional Networks

NetFlow / IPFIX
(IP Flow Information Export)

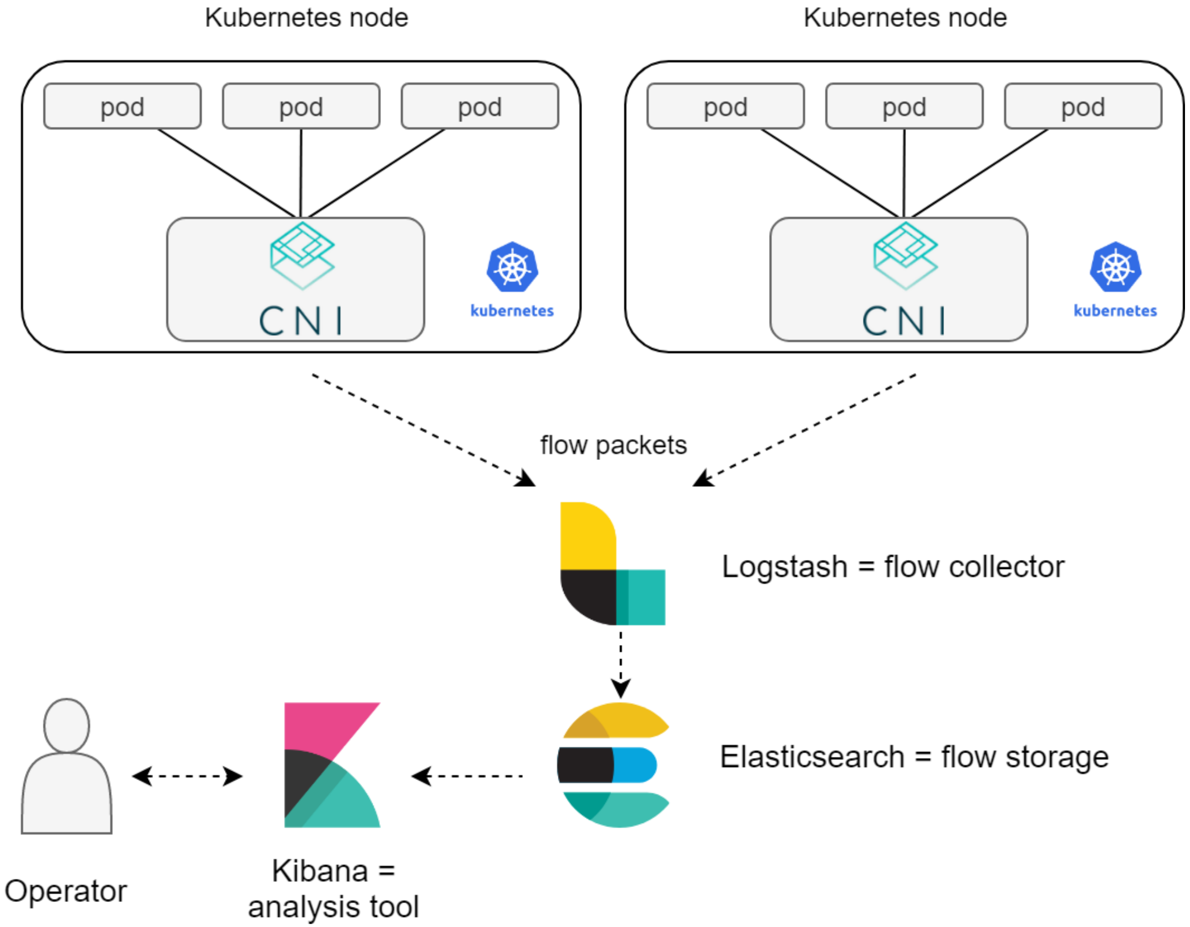
- Protocols for exporting information about each network conversation (flow)
- Flow: n-tuple: src/dst IP+port, IP protocol, ToS, interface, packet + data counts, timestamps, …
- Flow exporters: routers, switches, probes, other network devices
- Flow collector: reception, storage and pre-processing of flow data
- Analysis tool: analyzes received flow data



https://en.wikipedia.org/wiki/NetFlow

IPFIX in Kubernetes

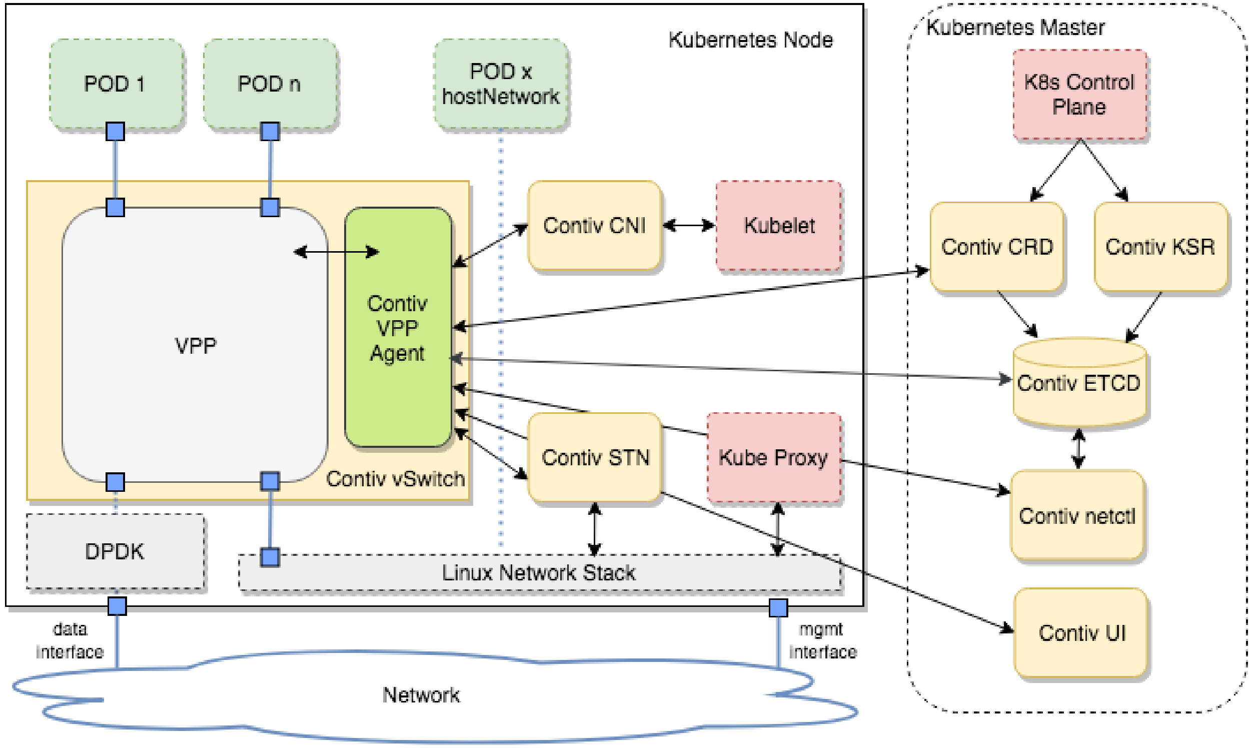
- Flow exporter: CNI plugin
  - CNI plugin acts as a router/switch between the pods
  - each CNI does the networking differently (e.g. multi-interface pods)
  - traffic is often encapsulated on the way between the nodes
- Cloud-native collector & analyzer can be built using the ELK stack:
  - flow collector: Logstash
  - flow storage: Elasticsearch
  - analysis tool: Kibana



Enabling IPFIX Export in Contiv-VPP CNI

Contiv-VPP (contivpp.io)

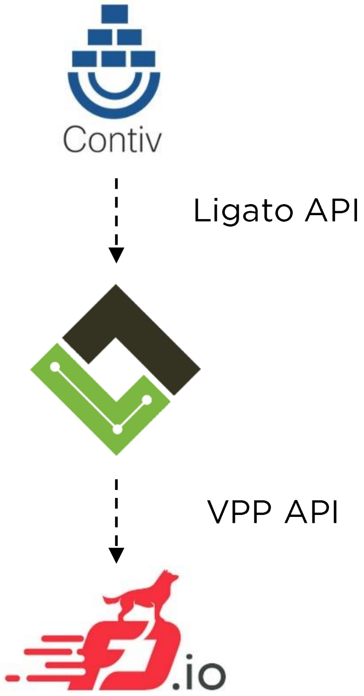
- CNI plugin based on FD.io VPP vSwitch (dataplane) running as a userspace process
- Focused on speed:
  - Vector Packet Processing
  - kube-proxy functionality in the userspace
  - memif interfaces
- Provides features aimed for CNFs (Cloud-Native Network Functions) deployments:
  - multiple pod interfaces
  - service function chaining between the pods
- VPP supports IPFIX - it just needs to be enabled



Enabling IPFIX Export in Contiv-VPP CNI

The Contiv-VPP CNI is modular and easily extendable. IPFIX support can be added by writing two tiny plugins:

- Contiv-VPP IPFIX plugin:
  - to enable IPFIX on each vSwitch (for each pod interface)
  - calls ligato.io API
- Ligato.io VPP Agent IPFIX plugin:
  - to enable IPFIX on VPP
  - calls VPP binary API via GoVPP
- FD.io VPP (data plane)
  - already contains IPFIX support
  - but if it was needed, it is extendable via plugins as well



Ligato API

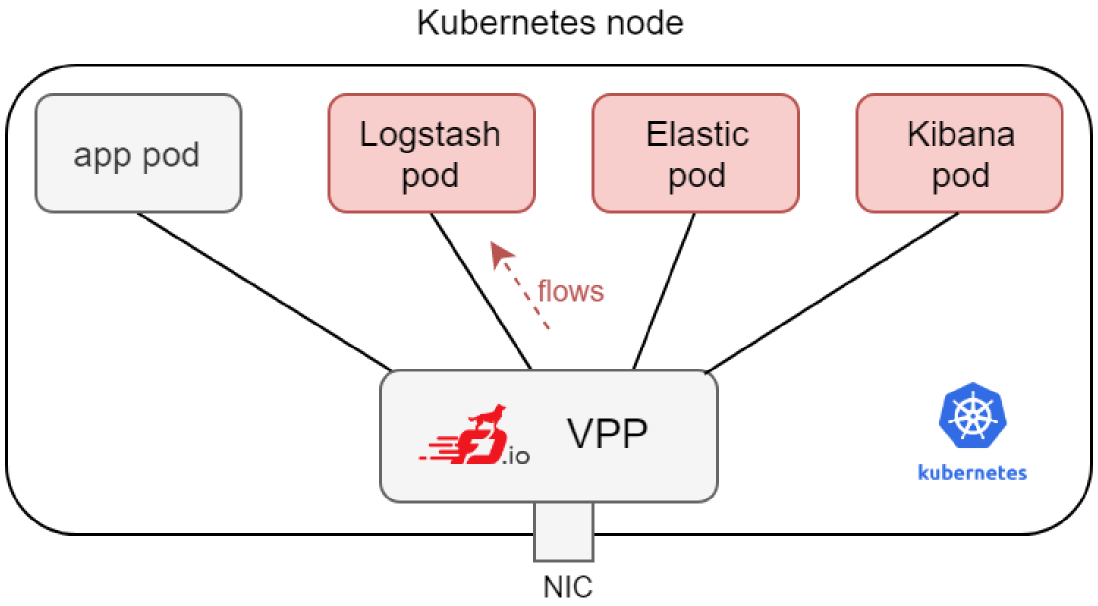
VPP API

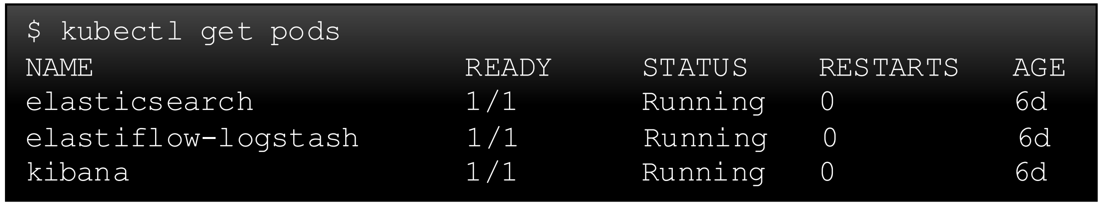
IPFIX Flow Collector & Analyzer based on ELK

github.com/robcowart/elastiflow:

- provides ready-to use ELK-based IPFIX collector & analyzer solution
- Logstash IPFIX/NetFlow/sFlow codec config & filters feeding Elasticsearch
- Kibana dashboards definitions

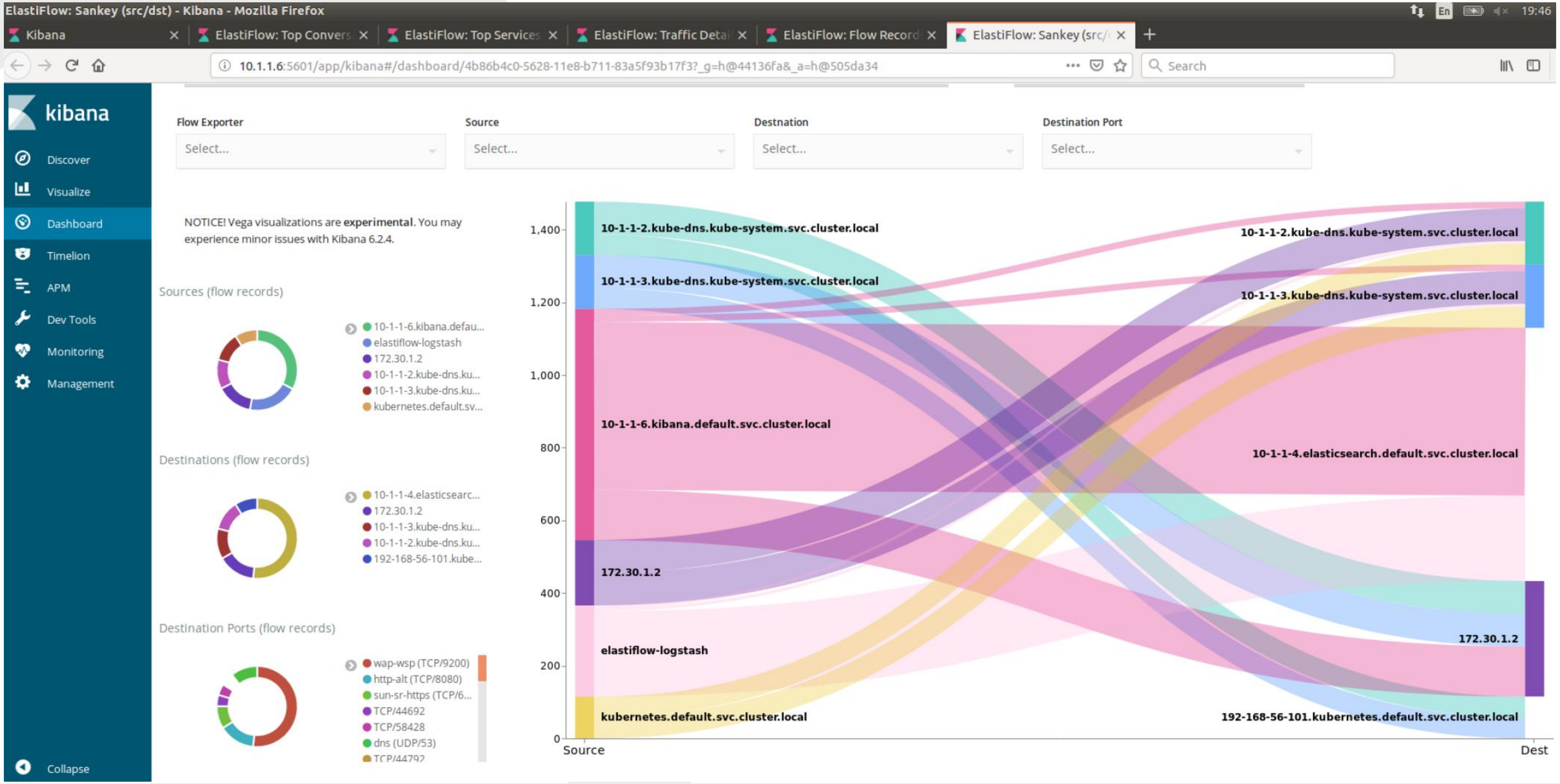
- Packaged into Docker containers and deployed in the K8s cluster
- Contiv-VPP CNI was configured to send the flow records into the Logstash pod





```
$ kubectl get pods
NAME                        READY     STATUS    RESTARTS   AGE
elasticsearch               1/1       Running   0          6d
elastiflow-logstash         1/1       Running   0          6d
kibana                      1/1       Running   0          6d
```

Kibana/ View on Traffic Flows Between Pods



ElastiFlow: Sankey (src/dst) - Kibana - Mozilla Firefox

10.1.1.6:5601/app/kibana#/dashboard/4b86b4c0-5628-11e8-b711-83a5f93b17f3?_g=h@44136fa&_a=h@505da34

Flow Exporter: Select...

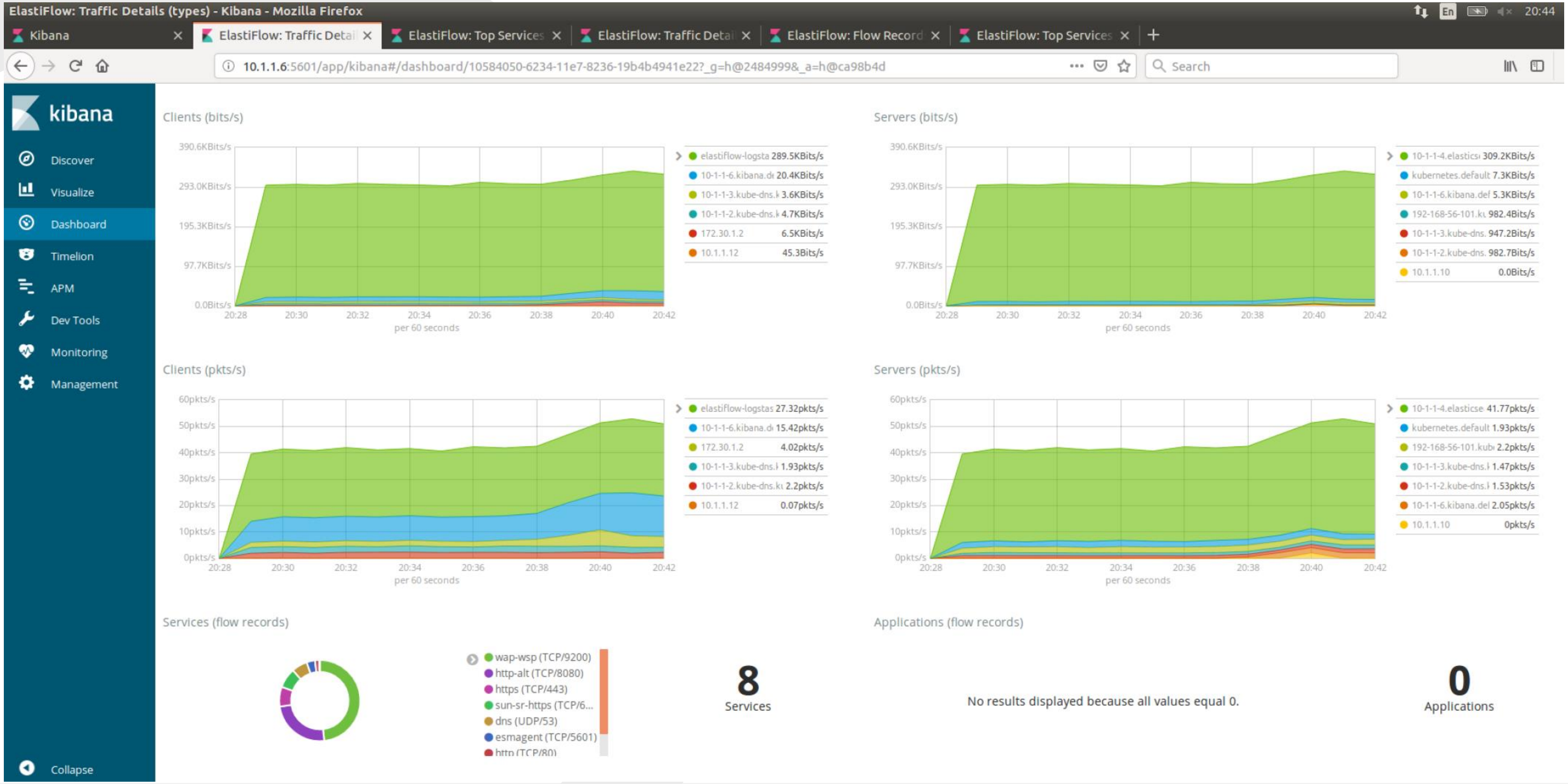
Source: Select...

Destnation: Select...

Destination Port: Select...

NOTICE! Vega visualizations are **experimental**. You may experience minor issues with Kibana 6.2.4.

Kibana/ Traffic Details

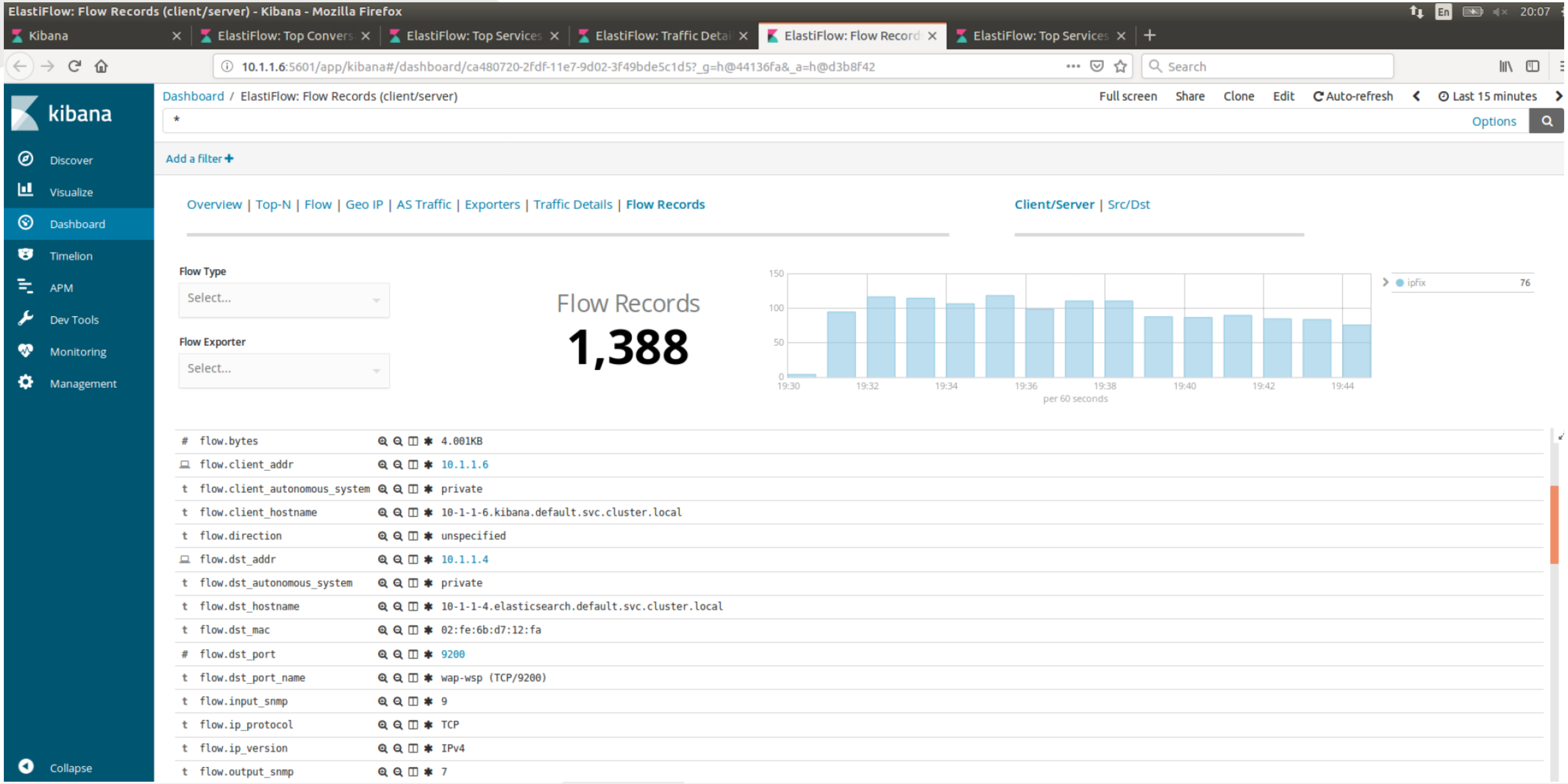


No results displayed because all values equal 0.

Network Flow Monitoring in K8s with Contiv-VPP CNI and Elastic Stack

PANTHEON .tech

Kibana/ Detailed Flow View



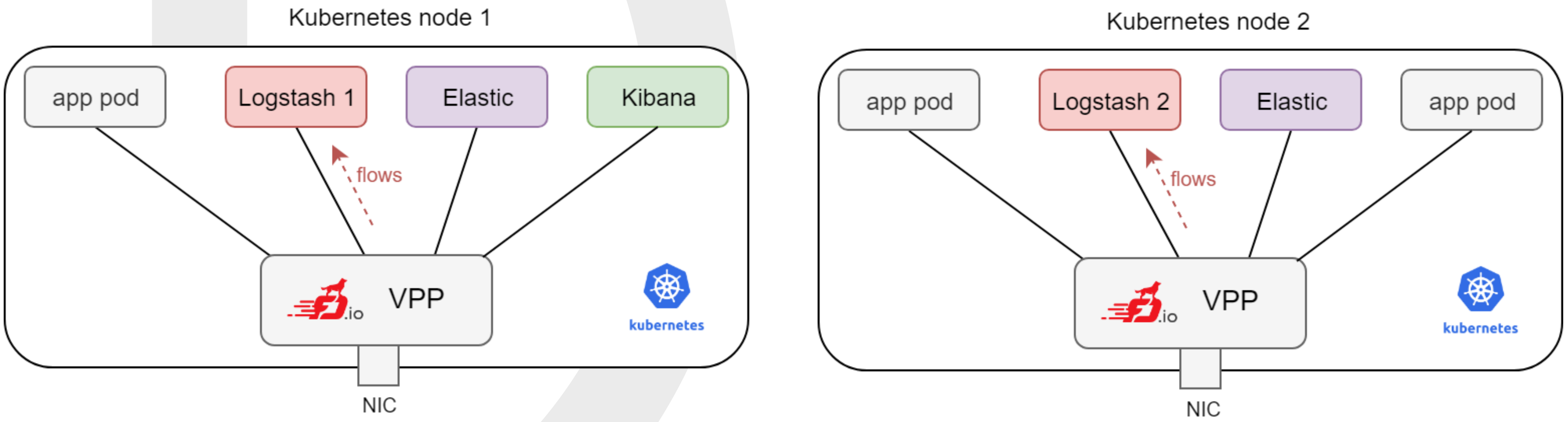
Flow Records

1,388

#	flow.bytes	4.001KB
💻	flow.client_addr	10.1.1.6
t	flow.client_autonomous_system	private
t	flow.client_hostname	10-1-1-6.kibana.default.svc.cluster.local
t	flow.direction	unspecified
💻	flow.dst_addr	10.1.1.4
t	flow.dst_autonomous_system	private
t	flow.dst_hostname	10-1-1-4.elasticsearch.default.svc.cluster.local
t	flow.dst_mac	02:fe:6b:d7:12:fa
#	flow.dst_port	9200
t	flow.dst_port_name	wap-wsp (TCP/9200)
t	flow.input_snmp	9
t	flow.ip_protocol	TCP
t	flow.ip_version	IPv4
t	flow.output_snmp	7

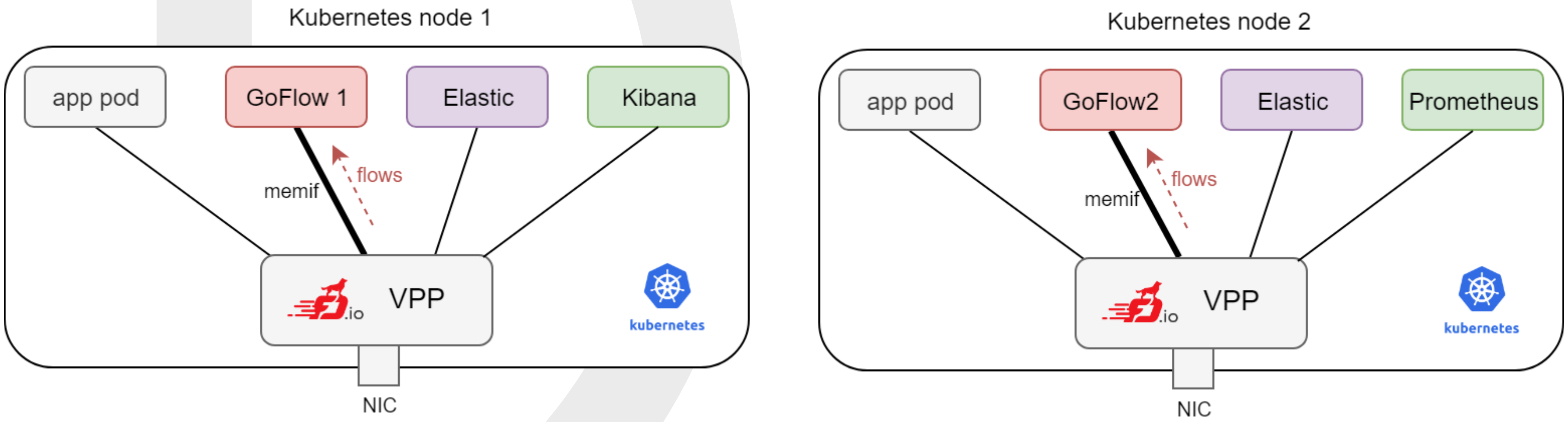
Possible Enhancements/ **Scaling**

- One Logstash pod on each node to keep the VPP-to-Logstash flow traffic within the same node
- Clustered Elastisearch deployment (covered by k8s service), ideally keep Logstash-to-Elastic traffic node-local as well
- One Kibana pod is enough (only a user interface)



Possible Enhancements/ **More Optimizations**

- Use memif between VPP and flow collector pod
- Use more lightweight flow collector (e.g. github.com/cloudflare/goflow), integrate with memif
- Add Elasticsearch source into Prometheus to provide more metrics



PANTHEON .tech

Thank You

Rastislav Szabo
rastislav.szabo@pantheon.tech
https://pantheon.tech