

# Packets Probes and eBPF Filtering in Skydive



# What is Skydive ?

Real-time network topology and protocol analyzer

Why: Network is complex (SDN)

- Topologies (physical, overlay, and application) are dynamic
- Multiple opaque tunneling layers
- Infrastructure complexity



# Use cases

## Network Topology Visualization (Discovery)

Skydive Analyzer 0.19.0-61badbcd4ab

**Metadata**

- Driver: veth
- EncapType: ether
- IPv4: 10.0.1.1/16
- IPv6: fe80::3c80:84ff:fe84:da1e/64
- IfIndex: 2
- MAC: 3e:80:84:84:da:1e
- MTU: 1500
- Name: veth0
- Neighbors: 5
- ParentIndex: 5
- PeerIndex: 5
- Speed: 10000
- State: UP
- TID: 0nda3a75-727e-574d-5ccd-25682b238644
- Type: veth

**Features**

**Metrics**

Total metrics				
Last	RxBytes	RxPackets	TxBytes	TxPackets
2:17:10 PM	11,676	190	936	12

Last metrics					
Last	Start	RxBytes	RxPackets	TxBytes	TxPackets
2:17:10 PM	2:17:05 PM	180	3		

Topology view  
Full  
Live

**Routing tables**

src: 10.0.1.1 (id: 255)

Prefix	NextHops	Protocol
10.0.0.0/32	IfIndex: 2	2
10.0.1.1/32	IfIndex: 2	2
10.0.255.255/32	IfIndex: 2	2
ff00::/8	IfIndex: 2	3
fe80::3c80:84ff:fe84:da1e/128	IfIndex: 2	2

src: 10.0.1.1

Prefix	NextHops
10.0.0.0/16	IfIndex: 2
fe80::/64	IfIndex: 2

**Features**

Feature	State
highdma	✓
rx-checksum	✓
rx-gro	✓
rx-vlan-hw-parse	✓
rx-vlan-stag-hw-parse	✓
tx-checksum-ip-generic	✓
tx-checksum-sctp	✓
tx-generic-segmentation	✓
tx-gre-csum-segmentation	✓
tx-gre-segmentation	✓
tx-ipip4-segmentation	✓
tx-ixp6-segmentation	✓
tx-lockless	✓
tx-scatter-gather	✓
tx-scatter-gather-fraglist	✓
tx-sctp-segmentation	✓
tx-tcp-segmentation	✓
tx-tcp-mangleid-segmentation	✓
tx-tcp-segmentation	✓
tx-tcp6-segmentation	✓
tx-udp-tnl-csum-segmentation	✓
tx-udp-tnl-segmentation	✓
tx-vlan-hw-insert	✓
tx-vlan-stag-hw-insert	✓





# Use cases

## Flow Troubleshooting (Tracking)

Skydive Analyzer 0.19.1-61badcbcd4ab

Preferences Documentation Status

ICMPv6	::	ff02::1:ffe6:69fa	1	0	86	0	0
ICMPv6	fe80::6c02:c1ff:fe1f:ddfa	ff02::16	5	0	470	0	0
ICMPv6	::	ff02::16	1	0	90	0	0
ICMPv6	fe80::586f:bdff:fee6:69fa	ff02::16	4	0	380	0	0
ICMPv6	::	ff02::16	1	0	90	0	0
ICMPv6	fe80::6c02:c1ff:fe1f:ddfa	ff02::2	6	0	420	0	0
ICMPv4	172.16.0.1	172.16.0.2	10	10	740	420	0.606

UUID : 6282609ea5e8a565  
LayersPath : Ethernet/IPv4/ICMPv4  
Application : ICMPv4

Link

Protocol : ETHERNET  
A : cs:d041:58:1c:73  
B : 76:a8:75:0a:b0:b8  
ID : 0

Network

Protocol : IPV4  
A : 172.16.0.1  
B : 172.16.0.2  
ID : 0

ICMP

Type : ECHO

Metric

ABPackets : 10  
ABBytes : 740 bytes  
BAPackets : 10  
BABBytes : 420 bytes  
Start : 28/09/2018, 12:43:45  
Last : 28/09/2018, 12:43:45

IPMetric

Fragments : 0  
FragmentErrors : 0

Start : 28/09/2018, 12:43:45  
Last : 28/09/2018, 12:43:45  
RTT : 0.606342 ms

TrackingID : e38c81871794612d  
L3TrackingID : 60a4a4d609433505  
ParentUUID :  
NodeTID : 9600086a-90d3-57de-75de-71f1f1b82528  
DevPacketCaptured : 0

Topology view  
Full  
Live



# Use cases

## Flow Monitoring (Grafana)





# Use cases

## Workflow (Validation, CI/CD, ...)

Captures Generator Flows Alerts Workflows

### Workflow

CheckConnectivity (Test connectivity between two interfaces)

### CheckConnectivity

#### Source node

3fdb53d8-e9c9-5c3b-5d4d-5e64a603708c

#### Destination node

5ac5ee86-8912-5875-73f1-440204e87619

Execute

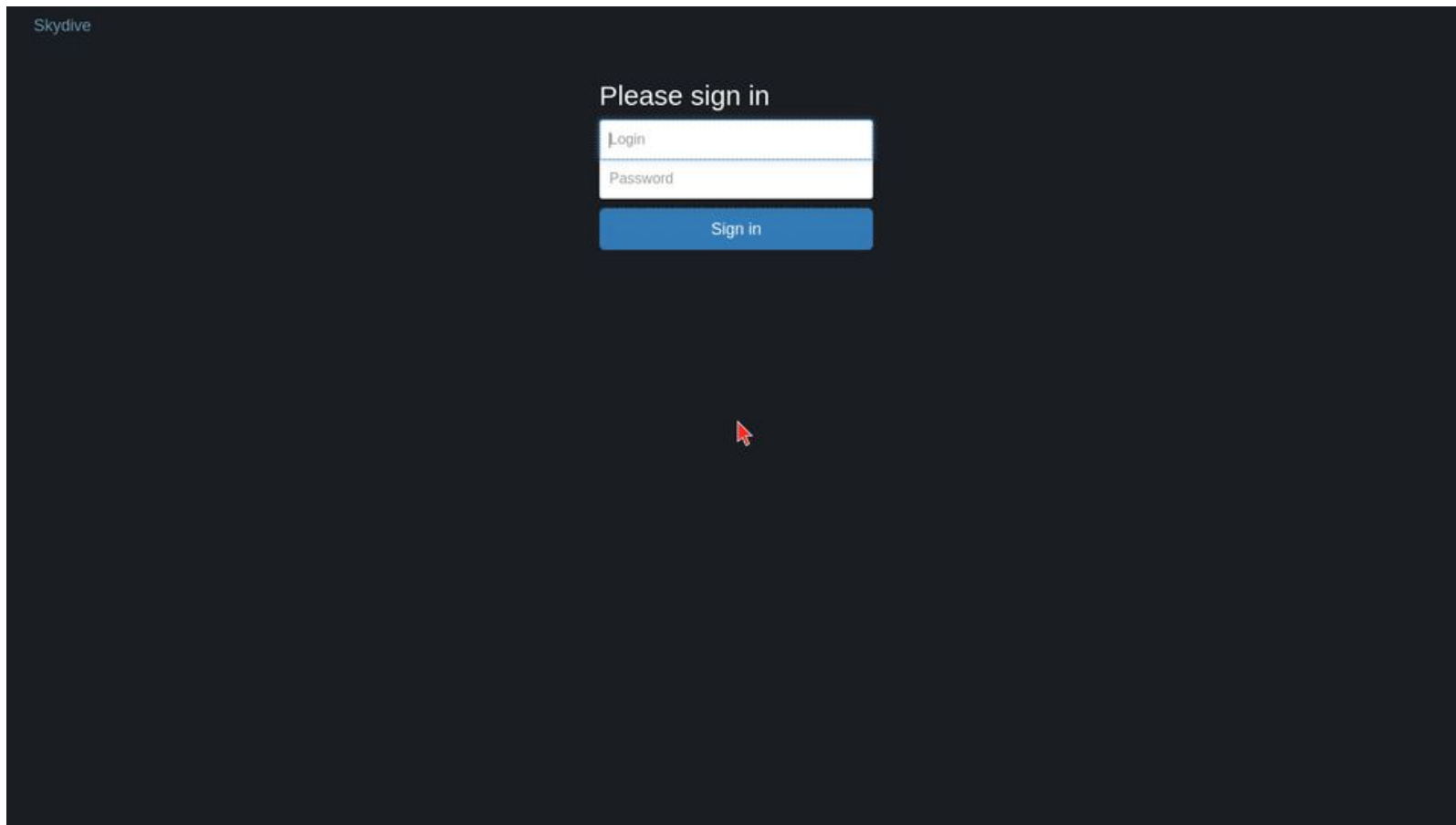
### Output

State : true  
Flows :

```
---  
name: "CheckConnectivity"  
description: "Test connectivity between two interfaces"  
parameters:  
  - name: source  
    description: Source node  
    type: node  
  - name: destination  
    description: Destination node  
    type: node  
source: |  
  function CheckConnectivity(from, to) {  
    var capture = new Capture();  
    capture.GremlinQuery = "G.V().Has('TID', ' + from + ').ShortestPathTo(Metadata('TID', ' + to + '))";  
  
    var packetInjection = new PacketInjection();  
    packetInjection.Src = "G.V().Has('TID', ' + from + ')"  
    packetInjection.Dst = "G.V().Has('TID', ' + to + ')"  
    packetInjection.Type = "icmp4"  
    packetInjection.ICMPPID = 123  
    packetInjection.Count = 5  
  
    return client.captures.create(capture).then(function (c) {  
      capture = c  
      return sleep(2000)  
    }).then(function () {  
      return client.packetInjections.create(packetInjection)  
    }).then(function () {  
      return sleep(2000)  
    }).then(function () {  
      return client.G.Flows().Has("ICMP.ID", 123)  
    }).then(function (flows) {  
      console.Log("Flows requested ! :-)")  
      console.Log(flows)  
      return { "State": true, "Flows": flows }  
    }).finally(function () {  
      client.captures.delete(capture.UUID)  
    })  
  }  
}
```



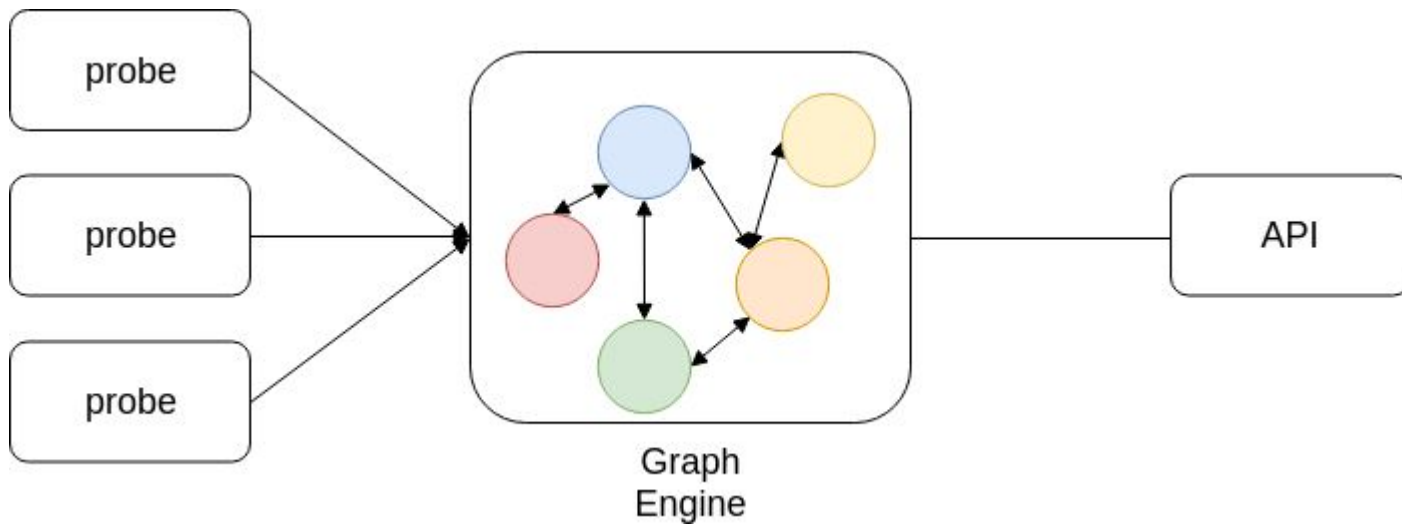
# Skydive in action







# Skydive Architecture





# Skydive Gremlin API

```
nplanel@t460np:~/devel/go/src/github.com/skydive-project/skydive$ skydive client query 'g.V().Has("Name", "eth0")'  
[  
  {  
    "ID": "c050a852-fbab-47b0-6087-62217ef295d8",  
    "Metadata": {  
      "Capture": {  
        "ID": "fc25caac-e34f-4fa8-5d1c-a5cb4fe7c7ae",  
        "PacketsDropped": 0,  
        "PacketsReceived": 64,  
        "State": "active"  
      },  
      "Driver": "veth",  
      "EncapType": "ether",  
      "IPV4": [  
        "172.16.0.1/24"  
      ],  
      "IPV6": [  
        "fe80::c4d0:f1ff:fe58:1c73/64"  
      ]  
    }  
  }  
]
```



# Use cases

## Flow Troubleshooting (Tracking)

Skydive Analyzer 0.19.1-61badcbcd4ab

Preferences Documentation Status

ICMPv6	::	ff02::1:ffe6:69fa	1	0	86	0	0
ICMPv6	fe80::6c02:c1ff:fe1f:ddfa	ff02::16	5	0	470	0	0
ICMPv6	::	ff02::16	1	0	90	0	0
ICMPv6	fe80::586f:bdff:fee6:69fa	ff02::16	4	0	380	0	0
ICMPv6	::	ff02::16	1	0	90	0	0
ICMPv6	fe80::6c02:c1ff:fe1f:ddfa	ff02::2	6	0	420	0	0
ICMPv4	172.16.0.1	172.16.0.2	10	10	740	420	0.606

**UUID** : 6282609ea5e8a565  
**LayersPath** : Ethernet/IPv4/ICMPv4  
**Application** : ICMPv4  
**Link**  
**Protocol** : ETHERNET  
**A** : c8:d0:11:58:1c:73  
**B** : 76:a8:75:0a:b0:b8  
**ID** : 0  
**Network**  
**Protocol** : IPV4  
**A** : 172.16.0.1  
**B** : 172.16.0.2  
**ID** : 0  
**ICMP**  
**Type** : ECHO  
**Metric**  
**APackets** : 10  
**CBytes** : 420 bytes  
**BAPackets** : 10  
**BABytes** : 420 bytes  
**Start** : 28/09/2018, 12:43:45  
**Last** : 28/09/2018, 12:43:45  
**IPMetric**  
**Fragments** : 0  
**FragmentErrors** : 0  
**Start** : 28/09/2018, 12:43:45  
**Last** : 28/09/2018, 12:43:45  
**RTT** : 0.606342 ms  
**TrackingID** : e38c81871794612d  
**L3TrackingID** : 60a4a4d609433505  
**ParentUUID** :  
**NodeTID** : 9600086a-90d3-57de-75de-71f1f1b82528  
**DevState** : Captured

G.Flows().Has('TrackingID', 'e38c81871794612d').Nodes().Dump()

Topology view  
Full  
Live



# Use cases

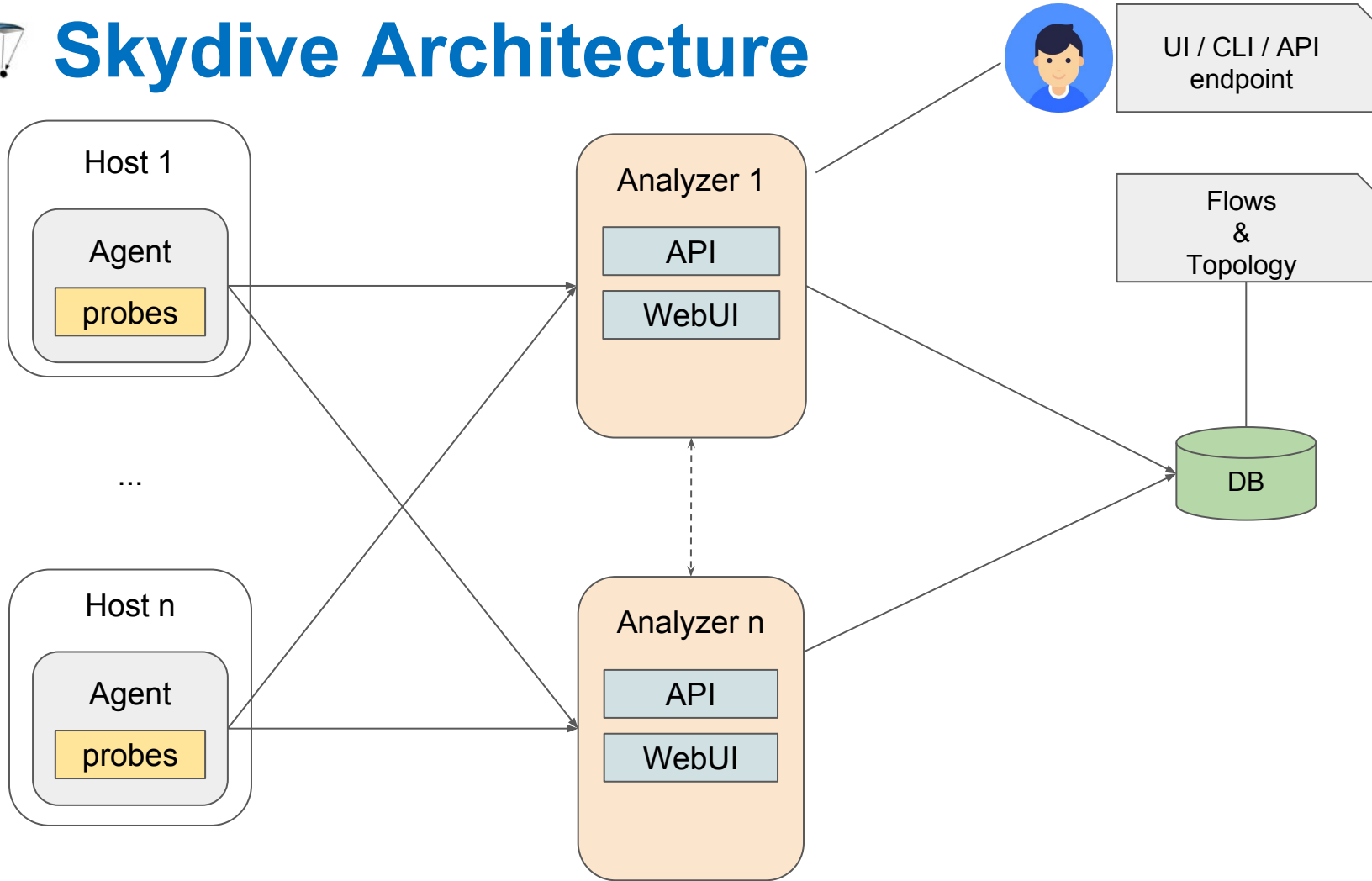
## Flow Monitoring (Grafana)



```
G.Flows().Has('Application', 'ICMPv4').Metrics().Aggregates()
```



# Skydive Architecture





# Topology Probes

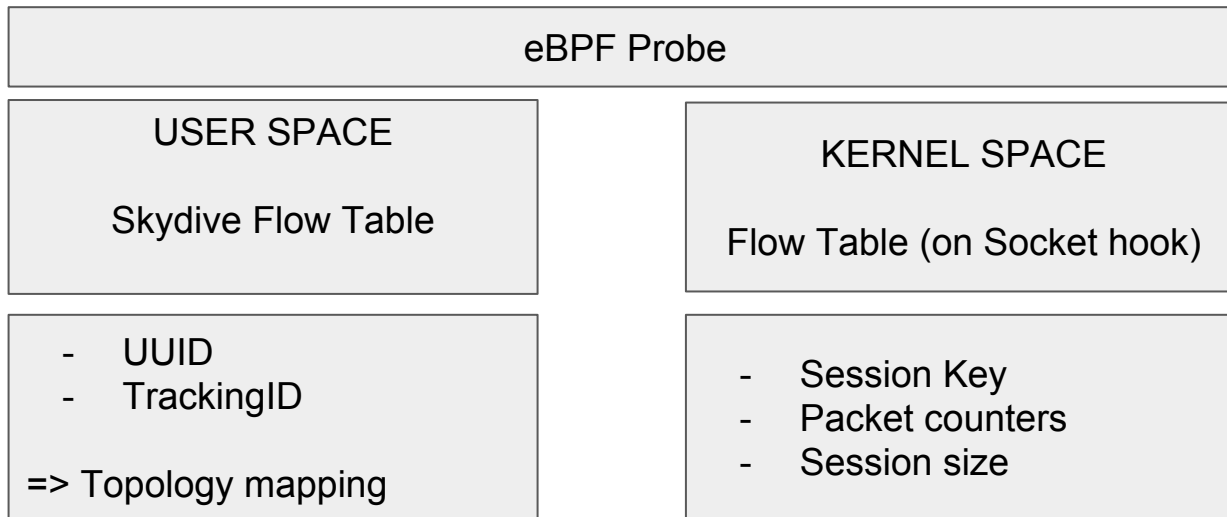
- Netlink
- NetNS
- OVS
- Docker
- Kubernetes
- Neutron
- SocketInfo



# Skydive capture probes

	Kernel	Overhead	Limitations
AF Packet	all	Huge	None
BPF (Syn/Fin/Rst/Ack/!Psh)	2.5+	Low	No packets metrics TCP only Start and end session
eBPF	3.18+	Low	No classification No fragmentation No tunneling (implementation)

# Skydive eBPF capture







# Skydive SocketInfo

The equivalent of “ss -tunp” saved in Skydive

```
udp      ESTAB      59136      0    192.168.78.45:43243    216.58.201.110:443
users: (("chrome",pid=1992,fd=465))
tcp      CLOSE-WAIT  32         0    192.168.78.45:43912    23.38.33.253:443
users: (("chrome",pid=1992,fd=515))
tcp      ESTAB      0          0    192.168.78.45:42058    192.168.78.38:22
users: (("ssh",pid=27194,fd=3))
...
```

## SocketInfo relies on Kprobes and eBPF

```
TCP,vm-2.rdodcloud,10.0.0.16,6633,/usr/bin/python2.7,neutron-openvsw,
=> vm-1.rdodcloud,10.0.0.15,/usr/sbin/ovs-vswitchd,ovs-vswitchd
```

```
TCP,vm-1.rdodcloud,10.0.0.15,3306,/usr/libexec/mysqld,mysqld,
=> vm-1.rdodcloud,10.0.0.15,/usr/bin/python2.7,nova-consoleaut
```

...



# Skydive Service/Application view

1 2 3 4 5 6 7 8 9 10 | VPN: None | VM: 0 | MEM: 2.3G | D: 32G | NET: +2.6K +2.7K | W: (92% Red Hat: Guest) 10.201.242.244 | E: down | 98% | 04/07/2018 18:37:20

localhost:8082/topology

Skydive Analyzer 0.18.0-3ffcd78a9d1 | Topology | Conversation | Discovery | Preferences | Documentation | Logout

Filter: G.V().Has('Name', regex('neutron.\*[ovsdb.\*']))  
Highlight: e.g. g.V().Has(,)

```
graph TD; ovsdb_server[ovsdb-server] --- neutron_dhcp[neutron-dhcp...]; ovsdb_server --- neutron_l3[neutron-l3-a...]; ovsdb_server --- ovsdb_client[ovsdb-client]; ovsdb_server --- neutron_open[neutron-open...];
```

Captures | Generator | Flows | Alerts | Workflows

Create

Metadata

Name: neutron-openvsw  
Type: service

Topology view  
G.V().Has('Name', regex('neutron.\*[ovsdb.\*']))  
Live



# Skydive roadmap

## Hybrid Capture

- Capture first packets until Classified (dissectors ?)
- Continue to capture flow counters via eBPF

## eBPF/Kprobe on local process

- Retrieve fragmentation / retransmission counters
- NetNS Create

## Extract graph engine (Steffi)

- Could be integrated in your tool



# Questions ?



<https://skydive.network>  
<https://github.com/skydive-project/skydive>