



CLOUDFLARE

Network Automation with Salt and NAPALM (or how we control 100's of PoPs around the world)

Mircea Ulinic
CloudFlare, London

RIPE 72 Copenhagen
May 2016

CloudFlare (a quick background)

- Once a website is part of the CloudFlare community, its web traffic is routed through our global network of 80+ locations
- How big?
 - Four+ million zones/domains
 - Authoritative for ~40% of Alexa top 1 million
 - 43+ billion DNS queries/day
 - Second only to Verisign
- 80+ anycast locations globally
 - 40 countries (and growing)
- Origin CA



Our big network challenges

- Deploy new PoPs
- Human error factor
- Replace equipment
- Monitor

Automation framework requirements

- Very scalable
- Concurrency
- Easily configurable & customizable
- Config verification & enforcement
- Periodically collect statistics
- Native caching and drivers for useful tools

Available solutions (most used)



Opinions

“The learning curve for Salt is higher and the intro docs are rough, but in the long-term **Salt’s docs are much better than Ansible’s, because they’re way more complete** (which is also why they’re much worse as intro docs).”

[Ryan D Lane](#)

“To me, Ansible was a great introduction to automated server configuration and deployment. Moving forward, **the scalability, speed and architecture of Salt has it going for it**. For cloud deployments I find the Salt architecture to be a better fit. I would not hesitate to use Salt in the future.”

[Jens Rantil](#)

Salt: the “unwanted child” of network automation

configuration agents, operators can affect configuration changes in a more programmatic way. Cisco provides various tools and frameworks to enable developers automate and program Nexus devices, including – NX-API REST (brings Model Driven Programmability (MDP) to standalone, Python, Puppet, Chef, Ansible etc.

<https://opennxos.cisco.com/public/getting-started>

Automation with Chef/Puppet and Ansible

by  [sachin vasudeva](#) on 10-21-2014 01:29 AM

<https://forums.juniper.net/t5/Automation-Programmability/Automation-with-Chef-Puppet-and-Ansible/ba-p/261773>

Why?

- Old references
- No feature for net devices as of yesterday
- Not well informed
- Not suitable for tiny VM networks

Salt at CloudFlare: used for years

Many thousands of servers already using Salt
Same tool for both servers and net devices

Salt (what fits the best our needs)	Ansible (most used in network automation)
• Long standing sessions • 20 types of modules • Customizable • Many thousands of CloudFlare servers • Comes embedded with features and tools • Native config enforcement logic • Real-time job • Job scheduling • Runner as a module • REST API • High Availability • GPG encryption • Pull from Git, SVN	• open/close session per module • 1 type of module • Customizable • ? • Need to install separate packages (“roles”) that are not necessarily dependent • Real-time job (Tower: \$\$) • Job Scheduling (Tower: \$\$) • Runner as a class • REST API (Ansible Tower: \$\$) • HA (Tower > Enterprise edition: \$\$\$\$) • Security (Tower: \$\$) • Pull from Git, SVN (Tower, \$\$)

Salt module types (selection)

- Execution modules
- Grains
- States
- Runners
- Pillars
- Returners



Embedded execution modules (selection)



redis



docker



SALTSTACK



slack



elastic



mongoDB®



PostgreSQL



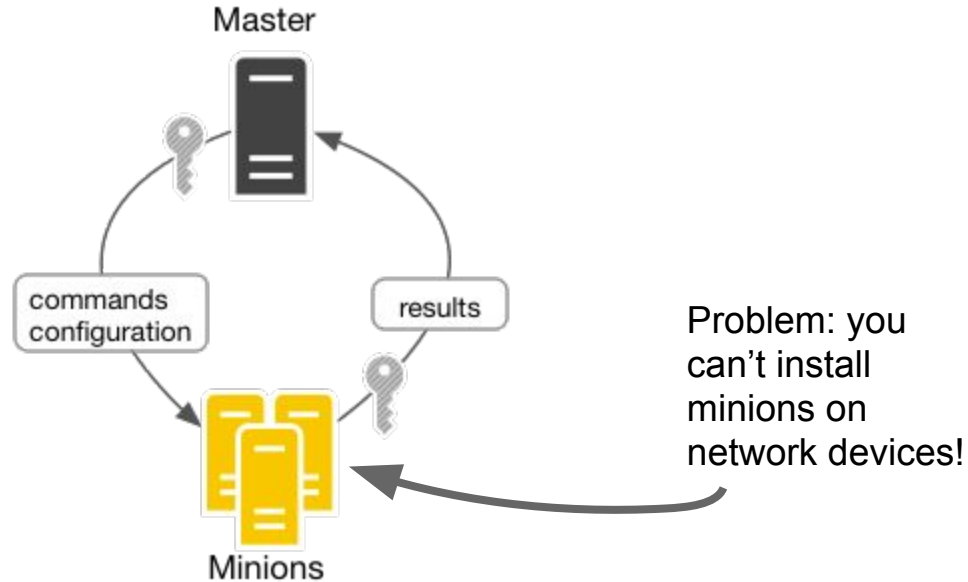
ORACLE®

Embedded returners (selection)



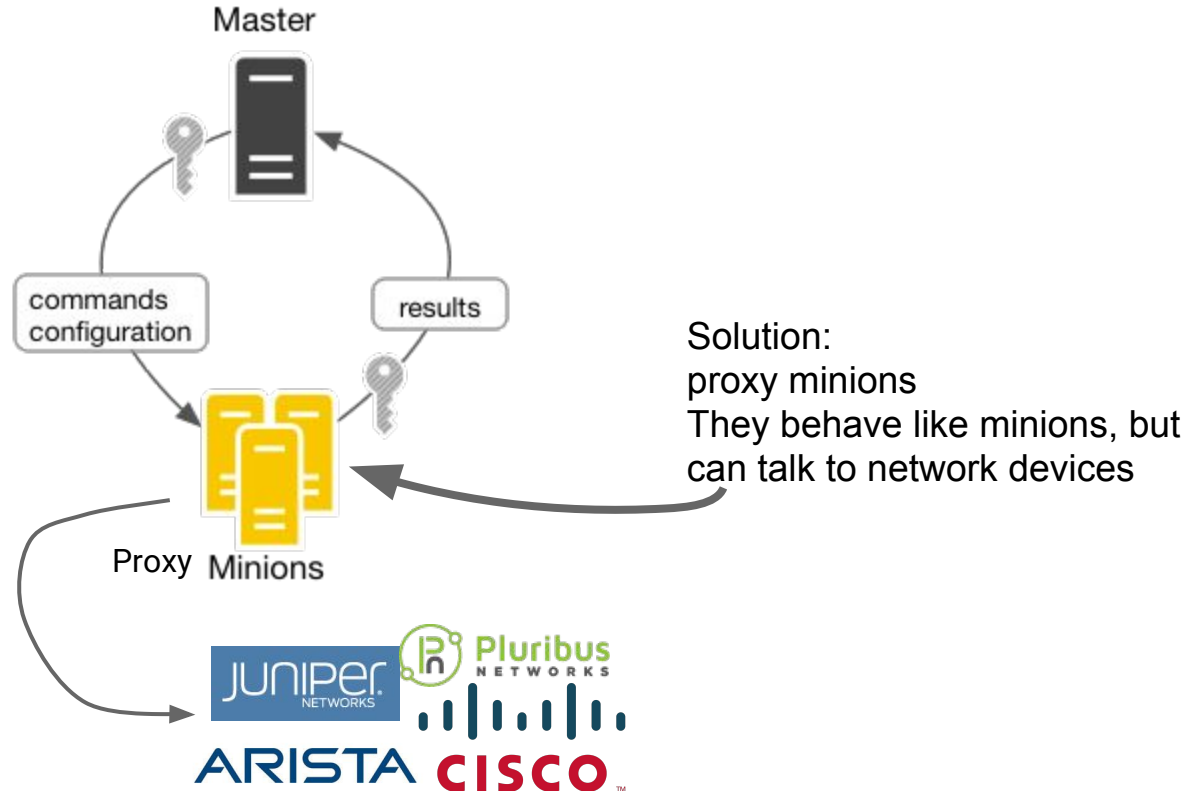
Easy to use: `salt edge05.cph01 net.facts --return sms`

Architecture



<https://www.digitalocean.com/community/tutorials/an-introduction-to-saltstack-terminology-and-concepts>

Proxy Minion

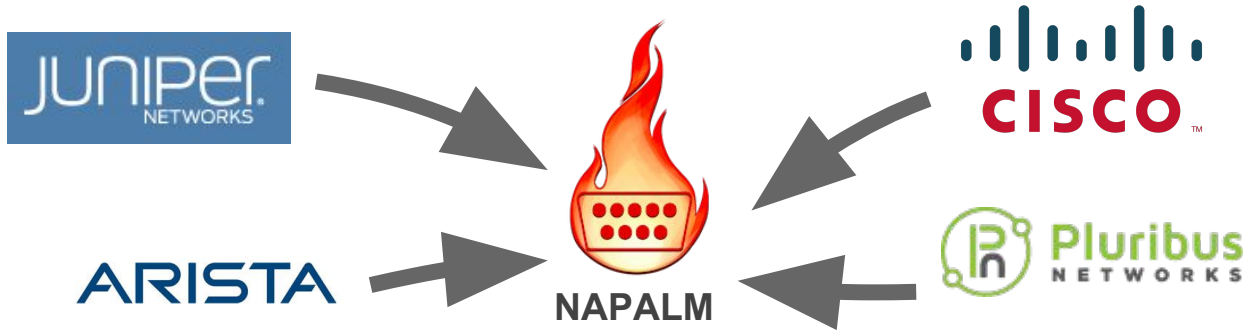


Disadvantages

- One proxy minion process / device
=> dedicated server preferred

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
18672	root	20	0	967M	98208	13400	S	0.0	0.1	0:43.14	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.prg01
19267	root	20	0	954M	81524	13176	S	0.0	0.1	0:47.88	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.mde01
18806	root	20	0	959M	86976	13136	S	0.0	0.1	0:48.55	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.kul01
19268	root	20	0	954M	81500	13344	S	0.0	0.1	0:44.57	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.pdx01
18772	root	20	0	965M	96228	13344	S	0.0	0.1	0:43.23	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.waw02
18794	root	20	0	956M	90192	13192	S	0.0	0.1	0:38.46	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.dme01
16573	root	20	0	957M	99M	13180	S	0.0	0.1	1:30.73	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.gru01
16638	root	20	0	802M	89644	10120	S	0.0	0.1	2:11.28	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.vie01
16803	root	20	0	802M	90032	10160	S	0.0	0.1	2:29.55	/usr/local/salt/virtualenv/bin/python2 /usr/local/salt/virtualenv/bin/salt-proxy --proxyid edge01.yyz01

NAPALM



(Network Automation and Programmability Abstraction Layer with Multivendor support)

<https://github.com/napalm-automation>

Fast growing library

February 2016

_	EOS	JunOS	IOS-XR	FortiOS	IBM	NXOS	IOS
get_facts	✓	✓	✓	✓	✗	✓	✓
get_interfaces	✓	✓	✓	✓	✗	✓	✓
get_ldp_neighbors	✓	✓	✓	✓	✗	✗	✓
get_bgp_neighbors	✓	✓	✓	✓	✗	✗	✓
get_environment	✓	✓	✓	✓	✗	✗	✓
get_interfaces_counters	✓	✓	✓	✓	✗	✗	✓

_	EOS	JunOS	IOS-XR	FortiOS	IBM	NXOS	IOS	Pluribus
cli	✓	✓	✓	✗	✗	✓	✓	✓
get_facts	✓	✓	✓	✓	✗	✓	✓	✓
get_environment	✓	✓	✓	✓	✗	✗	✓	✗
get_snmp_information	✓	✓	✓	✗	✗	✓	✓	✓
get_ntp_peers	✓	✓	✓	✗	✗	✓	✗	✓
get_ntp_stats	✓	✓	✓	✗	✗	✓	✓	✓
get_mac_address_table	✓	✓	✓	✗	✗	✓	✓	✓
get_arp_table	✓	✓	✓	✗	✗	✓	✓	✗
get_interfaces	✓	✓	✓	✓	✗	✓	✓	✓
get_interfaces_ip	✓	✓	✓	✗	✗	✓	✓	✗
get_ldp_neighbors	✓	✓	✓	✓	✗	✗	✓	✓
get_ldp_neighbors_detail	✓	✓	✓	✗	✗	✓	✓	✓
get_bgp_neighbors	✓	✓	✓	✓	✗	✗	✓	✗
get_bgp_neighbors_detail	✗	✓	✓	✗	✗	✗	✗	✗
get_bgp_config	✓	✓	✓	✗	✗	✗	✗	✗
get_route_to	✓	✓	✓	✗	✗	✗	✗	✗
get_probes_config	✗	✓	✓	✗	✗	✗	✗	✗
get_probes_results	✗	✓	✓	✗	✗	✗	✗	✗
get_users	✓	✓	✓	✗	✗	✓	✗	✓





Open source recipe: napalm-salt



NAPALM

NAPALM-Salt for Public use

- NAPALM integrated in Salt Carbon
- Execution Modules
 - [NET](#)
 - [BGP](#)
 - [NTP](#)
 - [Probes](#)
- States:
 - [NTP](#), [Probes](#)

NAPALM-Salt (examples):

1. salt "edge*" net.**traceroute** 8.8.8.8
2. salt -G "os:junos" net.**cli** "show version"
3. salt -C "sw* and G@os:nxos" net.**arp**
4. salt -G "os:iosxr and version:5.3.3" net.**mac**
5. salt -G "model:MX480" probes.**results**
6. salt -I "type:router" ntp.**set_peers** 10.1.130.10
10.1.130.18 10.1.130.22

Output example:

```
# salt --out=json edge05.cph01 net.arp
```

```
[
  {
    "interface": "ae2.100",
    "ip": "10.0.0.1",
    "mac": "00:0f:53:36:e4:50",
    "age": 129.0
  },
  {
    "interface": "xe-0/0/3.0",
    "ip": "10.0.0.2",
    "mac": "00:1d:70:83:40:c0",
    "age": 1101.0
  },
  {
    "interface": "xe-0/0/3.0",
    "ip": "10.0.0.3",
    "mac": "10:0e:7e:de:84:07",
    "age": 1276.0
  },
]
```

Abstracting configurations



Abstracted

```
protocols {
  bgp {
    group 4-PUBLIC-ANYCAST-PEERS {
      neighbor 192.168.0.1 {
        description "Amazon [WW HOSTING ANYCAST]";
        family inet {
          unicast {
            prefix-limit {
              maximum 500;
            }
          }
        }
        peer-as 16509;
      }
    }
  }
}
```

```
router bgp 13335
  neighbor 192.168.0.1
    remote-as 16509
    use neighbor-group 4-PUBLIC-ANYCAST-PEERS
    description "Amazon [WW HOSTING ANYCAST]"
    address-family ipv4 unicast
    maximum-prefix 500
```

```
bgp.neighbor:
  ip: 192.168.0.1
  group: 4-PUBLIC-ANYCAST-PEERS
  description: "Amazon [WW HOSTING ANYCAST]"
  remote_as: 16509
  prefix_limit: 500
```

Example

- Edge router with 1000 BGP peers
- Device is manufactured by *VendorA*
- Replaced by a device from *VendorB*

Most network engineers



Us

```
vi /etc/salt/pillar/edge05_cph01.sls
```

```
proxy:  
  driver: VendorA  
  proxytype: napalm  
  host: edge05.cph01  
  username: ripe  
  passwd: xxxx
```



```
proxy:  
  driver: VendorB  
  proxytype: napalm  
  host: edge05.cph01  
  username: ripe  
  passwd: xxxx
```

Maintain configuration updates

Define NTP peers in the Pillar

```
ntp.peers:  
- 10.1.130.22  
- 10.1.130.18  
- 10.1.128.10  
- 10.1.131.10  
- 10.1.132.10  
- 10.2.52.10  
- 10.2.48.10  
- 10.2.55.10  
- 10.2.50.10  
- 10.2.56.10
```



Schedule config enforcement checks

```
schedule:  
  ntp_config:  
    function: state.sls  
    args: router.ntp  
    returner: smtp  
    days: 1  
  bgp_config:  
    function: state.sls  
    args: router.bgp  
    hours: 2  
  probes_config:  
    function: state.sls  
    args: router.probes  
    days: 3  
  users_config:  
    function: state.sls  
    args: router.users  
    returner: hipchat  
    weeks: 1
```

NTP state output example

```
edge01.jnb01:
-----
      ID: ntp_config
      Function: netntp.managed
      Result: True
      Started: 09:50:41.228728
      Duration: 16813.319 ms
      Changes:
        -----
        peers:
          -----
          removed:
            - 10.10.1.1
        servers:
          -----
          added:
            - 17.xxx.xx.253
            - 40.xxx.xxx.7
          removed:
            - 83.xxx.xxx.118
            - 92.xx.xxx.58
            - 91.xx.xxx.42

Summary for edge01.jnb01
-----
Succeeded: 1 (changed=1)
Failed:    0
-----
Total states run:    1
Total run time: 16.813 s
```

What else can I do?

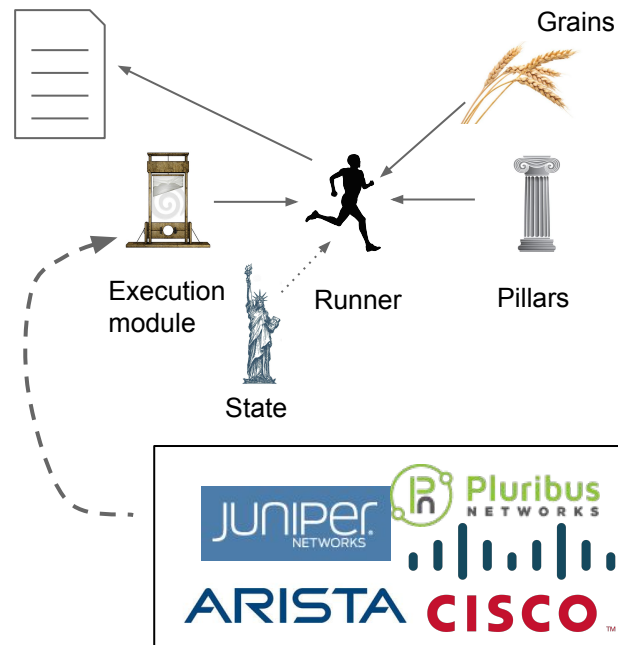
Examples:

Unique ASNs per geographic area

```
# salt-run bgp.asns_per_area

Canada : 96
Brazil : 167
Australia : 113
Peru : 4
USA : 410
Africa : 21
Asia : 362
Europe : 1004
North America : 421
South America : 183
Oceania : 162
Colombia : 5
Chile : 5
Argentina : 21

Execution time: 2.84680294991 s
#
```



Find stuff (using Salt mine)

```
# salt-run net.find core01.sjc01
Pattern "core01.sjc01" found in the description of the following interfaces
=====
| Device | Interface | Interface Description | UP | Enabled | Speed [Mbps] | MAC Address | IP Addresses |
=====
| sw01.sjc01 | ae0 | core01.sjc01 | True | True | 40000 | 78:fe:3d:ed:02:83 | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| sw01.sjc01 | xe-1/1/0 | ae0:core01.sjc01:Et3/2/3 | True | True | 10000 | 78:fe:3d:ed:02:83 | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| sw01.sjc01 | xe-1/1/1 | ae0:core01.sjc01:Et3/2/4 | True | True | 10000 | 78:fe:3d:ed:02:83 | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| sw01.sjc01 | xe-0/1/1 | ae0:core01.sjc01:Et3/2/2 | True | True | 10000 | 78:fe:3d:ed:02:83 | |
|-----|-----|-----|-----|-----|-----|-----|-----|

# salt-run net.find 54:e0:32:7e:85:2d
Details for interface xe-4/0/5 on device edge01.sjc01
=====
| Device | Interface | Interface Description | UP | Enabled | Speed [Mbps] | MAC Address | IP Addresses |
=====
| edge01.sjc01 | xe-4/0/5 | | | True | 10000 | 54:e0:32:7e:85:2d | |
|-----|-----|-----|-----|-----|-----|-----|-----|

# salt-run net.find 00:0f:53:36:e4:50
Found ARP entry on edge05.cph01: 10.0.0.1 <-> 00:0F:53:36:E4:50
```

BGP neighbors of some ASNs

```
# salt-run bgp.neighbors 15169 16509 32934 13414
```

BGP Neighbors for 15169, 16509, 32934, 13414:

Device	As Number	Neighbor Address	State	#Active/Received/Accepted/Damped	Policy In
edge01.dub01	15169	20	Established	27/48/48/0	6-PUBLIC-PEER-IN
edge01.dub01	16509	20	Established	1/1/1/0	6-PUBLIC-PEER-IN
edge01.nrt01	13414	20	Established	59/59/59/0	4-PUBLIC-PEER-IN
edge01.nrt01	13414	20	Established	3/3/3/0	6-PUBLIC-PEER-IN
edge01.nrt01	16509	20	Established	71/71/71/0	4-PUBLIC-PEER-IN
edge01.nrt01	16509	20	Established	1/1/1/0	6-PUBLIC-PEER-IN
edge01.nrt01	32934	20	Established	26/26/26/0	4-PUBLIC-PEER-IN
edge01.nrt01	32934	20	Established	14/15/14/0	6-PUBLIC-PEER-IN
edge01.nrt01	15169	20	Established	331/331/331/0	4-PUBLIC-PEER-IN
edge01.tpe01	15169	20	Established	331/331/331/0	4-PUBLIC-PEER-IN
edge01.tpe01	15169	24	Established	48/48/48/0	6-PUBLIC-PEER-IN
edge01.waw02	16509	19	Established	5/5/5/0	4-PUBLIC-PEER-IN
edge01.waw02	15169	19	Established	177/331/331/0	4-PUBLIC-PEER-IN
edge01.waw02	15169	20	Established	22/48/48/0	6-PUBLIC-PEER-IN
edge01.waw02	32934	21	Established	26/26/26/0	4-PUBLIC-PEER-IN
edge01.waw02	32934	20	Established	14/14/14/0	6-PUBLIC-PEER-IN
edge01.lhr01	13414	19	Established	59/59/59/0	4-PUBLIC-PEER-IN
edge01.lhr01	16509	20	Established	0/1/0/0	REJECT-ALL
edge01.gru01	32934	20	Established	12/12/12/0	6-PUBLIC-PEER-IN

Monitor your network

```
# Redis details:
redis.host: localhost
redis.port: 6379

# Schedulers
schedule:
  traceroute_runner:
    function: traceroute.collect
    hours: 2
```



```
2071) "traceroute:edge01.sjc01-edge01.lhr01-Tata-4"
2072) "traceroute:edge01.iad02-edge01.sjc01-GTT-4"
2074) "traceroute:edge01.fra03-edge01.sea01-Cogent-4"
2075) "traceroute:edge01.yul01-edge01.lax01-Cogent-4"
2076) "traceroute:edge01.zrh01-edge01.fra03-GTT-4"
2077) "traceroute:edge01.mxp01-edge01.ams01-GTT-4"
2078) "traceroute:edge01.mia01-edge01.lhr01-GTT-4"
2079) "traceroute:edge01.msp01-edge01.scl01-Telefonica-4"
2080) "traceroute:edge01.fra03-edge01.mia01-Telia-4"
2081) "traceroute:edge01.lim01-edge01.scl01-Telefonica-4"
2082) "traceroute:edge01.arn01-edge01.mia01-GTT-4"
2083) "traceroute:edge01.prg01-edge01.lax01-GTT-4"
2084) "traceroute:edge01.osl01-edge01.lhr01-GTT-4"
```

Traceroute diff

Current:

time	src		dst		probe loss		
10:22:46 14-05-16	1.1.1.1		2.2.2.2		26		
	edge01.phx01		edge01.lax01				

hop	rtt 1	rtt 2	rtt 3	ip	host	asn	asn description
1	29.663	29.705	30.057		be2929.ccr21.phx02		
2	41.987				be2932.ccr22.lax01		
		42.604	41.051		be2931.ccr21.lax01		
3	41.912		42.036		be2179.ccr23.lax05		
		41.685		2.2.2.2	be2180.ccr23.lax05		
4	66.714	66.504	66.329		2.2.2.2		

Previous:

time	src		dst		probe loss		
08:32:15 14-05-16	1.1.1.1		2.2.2.2		0		

hop	rtt 1	rtt 2	rtt 3	ip	host	asn	asn description
1	29.71				be2929.ccr21.phx02		
		30.569	30.092		be2930.ccr22.phx02		
2	41.453	43.002			be2931.ccr21.lax01		
			41.272		be2932.ccr22.lax01		
3	43.856			2.2.2.2	be2180.ccr23.lax05		
		42.465	41.741		be2179.ccr23.lax05		
4	41.433	42.812	41.479	2.2.2.2	2.2.2.2		

How can you use it?

```
# apt-get install salt-master (install guide)
```

```
# pip install napalm
```

Examples:

<https://github.com/napalm-automation/napalm-salt>

How can you contribute?

- NAPALM Automation:
<https://github.com/napalm-automation>
- SaltStack
<https://github.com/saltstack/salt>

Need help/advice?

Join [#saltstack #napalm](https://networktoencode.herokuapp.com/rooms)

By email:

- Mircea Ulinic: mircea@cloudflare.com
- Jerome Fleury: jf@cloudflare.com

Questions



By email:

- Mircea Ulinic: mircea@cloudflare.com
- Jerome Fleury: jf@cloudflare.com