Monitoring & Network Management Systems

System and Network Administration

2024/25

Pierre-Philipp Braun <pbraun@nethence.com>

Table of contents

Incident Monitoring (status alerts)

```
\|/          (__)
     `\------(oo)
       ||    (__)
       ||w--||     \|/
   \|/
```

What makes the difference between a company that has incidents every day and those who don’t?…

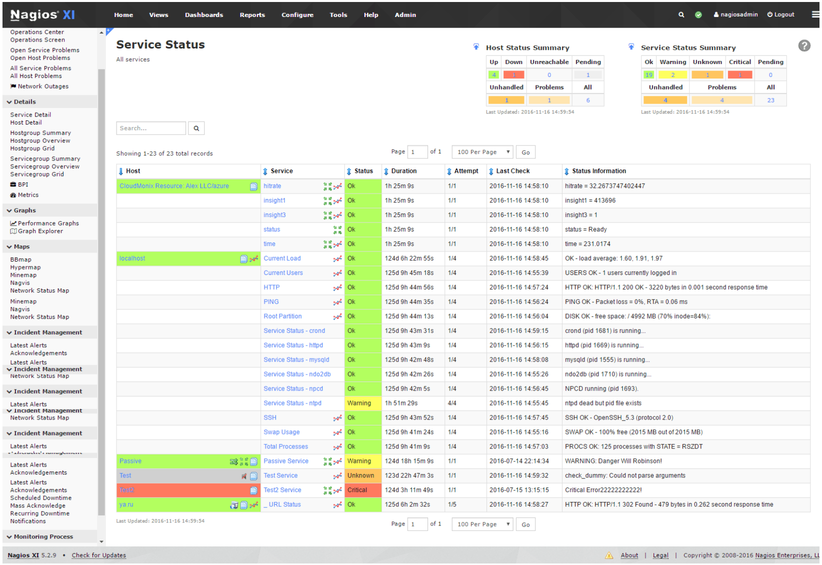
==> Network Operations Center (NOC) team with monitoring dashboards

- they know who to call
- they are been told what incidents are ok to stay

If no budget for a NOC, setup alerts by email or SMS (OVH and IPPI offers relays).

THE DASH-BOARD

- big screen(s) in operations room
- view alerts live on dashboard
- view alerts live on detailled host/services (Nagios)
- view performance graphs live on customized dashboards (Zabbix)



Service Status

All services

Host Status Summary

Up	Down	Unreachable	Pending
4	1	0	1

Unhandled	Problems	All
1	1	6

Last Updated: 2016-11-16 14:59:54

Service Status Summary

Ok	Warning	Unknown	Critical	Pending
19	2	1	1	0

Unhandled	Problems	All
4	4	23

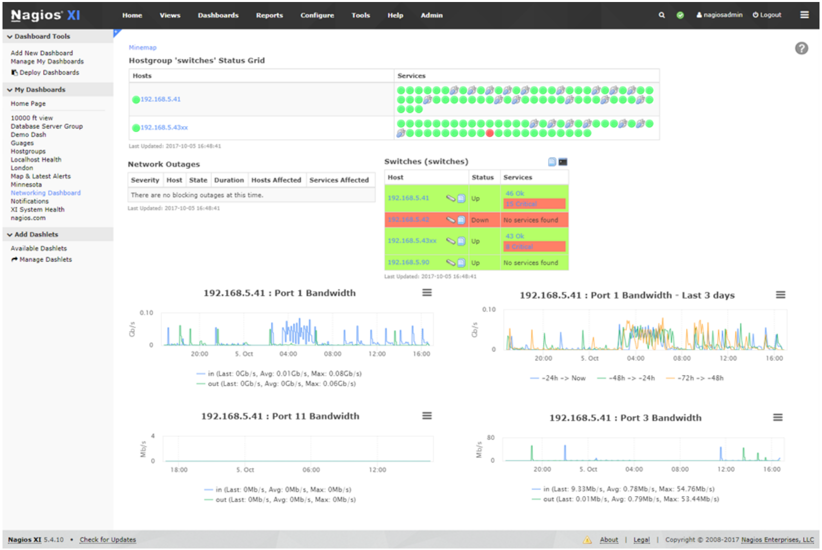
Last Updated: 2016-11-16 14:59:54

Showing 1-23 of 23 total records

Host	Service	Status	Duration	Attempt	Last Check	Status Information
CloudMonix Resource: Alex LLC/azure	hitrate	Ok	1h 25m 9s	1/1	2016-11-16 14:58:10	hitrate = 32.2673747402447
	insight1	Ok	1h 25m 9s	1/1	2016-11-16 14:58:10	insight1 = 413696
	insight3	Ok	1h 25m 9s	1/1	2016-11-16 14:58:10	insight3 = 1
	status	Ok	1h 25m 9s	1/1	2016-11-16 14:58:10	status = Ready
	time	Ok	1h 25m 9s	1/1	2016-11-16 14:58:10	time = 231.0174
localhost	Current Load	Ok	124d 6h 22m 55s	1/4	2016-11-16 14:58:45	OK - load average: 1.60, 1.91, 1.97
	Current Users	Ok	125d 9h 45m 18s	1/4	2016-11-16 14:55:39	USERS OK - 1 users currently logged in
	HTTP	Ok	125d 9h 44m 56s	1/4	2016-11-16 14:57:24	HTTP OK: HTTP/1.1 200 OK - 3220 bytes in 0.001 second response time
	PING	Ok	125d 9h 44m 35s	1/4	2016-11-16 14:56:24	PING OK - Packet loss = 0%, RTA = 0.06 ms
	Root Partition	Ok	125d 9h 44m 13s	1/4	2016-11-16 14:56:04	DISK OK - free space: / 4992 MB (70% inode=84%):
	Service Status - crond	Ok	125d 9h 43m 31s	1/4	2016-11-16 14:59:15	crond (pid 1681) is running...
	Service Status - httpd	Ok	125d 9h 43m 9s	1/4	2016-11-16 14:56:15	httpd (pid 1669) is running...
	Service Status - mysqld	Ok	125d 9h 42m 48s	1/4	2016-11-16 14:58:08	mysqld (pid 1555) is running...
	Service Status - ndo2db	Ok	125d 9h 42m 26s	1/4	2016-11-16 14:55:26	ndo2db (pid 1710) is running...
	Service Status - npcd	Ok	125d 9h 42m 5s	1/4	2016-11-16 14:56:45	NPCD running (pid 1693).
	Service Status - ntpd	Warning	1h 51m 29s	4/4	2016-11-16 14:55:45	ntpd dead but pid file exists
	SSH	Ok	125d 9h 43m 52s	1/4	2016-11-16 14:57:45	SSH OK - OpenSSH_5.3 (protocol 2.0)
	Swap Usage	Ok	125d 9h 41m 24s	1/4	2016-11-16 14:55:16	SWAP OK - 100% free (2015 MB out of 2015 MB)
	Total Processes	Ok	125d 9h 41m 9s	1/4	2016-11-16 14:57:03	PROCS OK: 125 processes with STATE = RSZDT
Passive	Passive Service	Warning	124d 18h 15m 9s	1/1	2016-07-14 22:14:34	WARNING: Danger Will Robinson!
Test	Test Service	Unknown	123d 22h 47m 3s	1/1	2016-11-16 14:59:32	check_dummy: Could not parse arguments
Test2	Test2 Service	Critical	124d 3h 11m 49s	1/1	2016-07-15 13:15:15	Critical Error22222222222!
ya.ru	_URL Status	Ok	125d 6h 2m 32s	1/5	2016-11-16 14:58:27	HTTP OK: HTTP/1.1 302 Found - 479 bytes in 0.262 second response time

Last Updated: 2016-11-16 14:59:54

Nagios XI host/services



Hostgroup 'switches' Status Grid

Hosts	Services
● 192.168.5.41	
● 192.168.5.43xx	

Last Updated: 2017-10-05 16:48:41

Network Outages

Severity	Host	State	Duration	Hosts Affected	Services Affected
There are no blocking outages at this time.					

Last Updated: 2017-10-05 16:48:41

Switches (switches)

Host	Status	Services
192.168.5.41	Up	46 Ok 15 Critical
192.168.5.42	Down	No services found
192.168.5.43xx	Up	43 Ok 8 Critical
192.168.5.90	Up	No services found

Last Updated: 2017-10-05 16:48:41

192.168.5.41 : Port 1 Bandwidth
Gb/s
0.10
0
20:00
5. Oct
04:00
08:00
12:00
16:00
in (Last: 0Gb/s, Avg: 0.01Gb/s, Max: 0.08Gb/s)
out (Last: 0Gb/s, Avg: 0Gb/s, Max: 0.06Gb/s)
192.168.5.41 : Port 1 Bandwidth - Last 3 days
Gb/s
0.10
0
20:00
5. Oct
04:00
08:00
12:00
16:00
-24h -> Now
-48h -> -24h
-72h -> -48h
192.168.5.41 : Port 11 Bandwidth
Mb/s
4
0
18:00
5. Oct
06:00
12:00
in (Last: 0Mb/s, Avg: 0Mb/s, Max: 0Mb/s)
out (Last: 0Mb/s, Avg: 0Mb/s, Max: 0Mb/s)
192.168.5.41 : Port 3 Bandwidth
Mb/s
80
0
20:00
5. Oct
04:00
08:00
12:00
16:00
in (Last: 9.33Mb/s, Avg: 0.78Mb/s, Max: 54.76Mb/s)
out (Last: 0.01Mb/s, Avg: 0.79Mb/s, Max: 53.44Mb/s)
Nagios XI 5.4.10 • Check for Updates
About | Legal | Copyright © 2008-2017 Nagios Enterprises, LLC

Nagios XI hostgroups

TYPES OF CHECKS

- Remote/network checks & metrics
- Local/agent checks & metrics
- Hypervisor/host metrics
- BMC
- SNMP

REMOTE ALERTS

Viewing and receiving alerts on ***status vs. static thresholds***

- Host absence (no ping response)
- Services down
- Services too slow
- Web pages down
- Web pages too slow

ADVANCED & DEEP-DIVE SYSTEM ALERTS

SYSTEM/VMM

- RAID status by means of e.g. HPE cciss userland tools
- NIC negociated speed e.g. `1000baseT-FD`
- LACP…
- File-system usage e.g. close to 90%

and eventually

- Motherboard and disks' temperature
- Fan RPM

BMC ALERTS

BMC

- RAID status directly by BMC?
- Chassis temperature
- Fan statuses and RPM
- Energy-waste (Watt / Voltages)

Viewing and receiving alerts on ***timed thresholds***

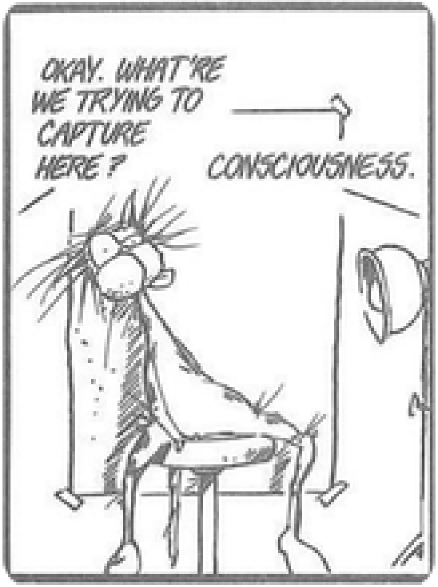
VMM performance bottlenecks

- Constant CPU 100%
- Constant RAM 100%
- Constant DISK I/O 100%
- Bandwidth usage
    - Per network link RX 100% during 15 minutes…
    - Per network link TX 100% during 2,5 hours…

About network *TX* overload, that should rather be for IDS/IPS data leak prevention.

SNMP ALERTS

covered in another lecture: SNE/NETWORK/SNMP



source: acme.com

FROM-THE-DIY-DEPT

- Nagios plug-ins not that hard – shell scripts with output and exit codes
- DIY alerting with SSH

For example, how to check file-system usage manually?...

==> File-system usage

```
df -hT
```

```
slack2# df -P
Filesystem    1024-blocks    Used Available Capacity Mounted on
/dev/root         69075456 55685528   9858000      85% /
devtmpfs          64948268        0  64948268       0% /dev
tmpfs             64951772      900  64950872       1% /run
tmpfs             64951772        0  64951772       0% /dev/shm
cgroup_root     64951772        0  64951772      0% /sys/fs/cgroup
/dev/sdb1        313296192 97542500 200061100      33% /data
cgmfs                100       0      100      0% /run/cgmanager/fs
```

Note the flag to standardize things across platforms

```
   -P, --portability
          use the POSIX output format
```

DIY alerting - file-system usage

Prints output only if there is a problem…

```
vi /root/report/diskusage.bash

#!/bin/bash

tmp=`df -P | sed 1d | grep -vE '^udev|tmpfs|^cgroup|^rpool/ROOT/'`

echo "$tmp" | while read line; do
        percent=`echo $line | awk '{print $5}' | sed 's/%//'`
        (( percent > 89 )) && echo $line
        unset percent
done; unset line

chmod +x /root/report/diskusage.bash
```

Now how and where to execute that?...

(and at what frequency?...)

==> ClusterIt (DSH) as cron jobs

May be executed in a loop for live display -or- put it in a cron job

```
crontab -e


*/5 * * * * /usr/pkg/bin/dsh -e -g linux -s /root/report/diskusage.bash
```

Note jobs can be scheduled from the backup server (which may have all the necessary SSH accesses already)

Scaleway Status
<https://status.scaleway.com/>

```
\|/          (__)
     `\------(oo)
       ||    (__)
       ||w--||     \|/
   \|/
```

// Questions on incident monitoring?

Performance Tools & Graphs

What methods would you consider to keep track of hardware resource usage and performance?

Advantages

- performance graphs for daily activity
  - spot misbehaving nodes & services (trigger alerts)
  - predict incidents
  - catch the DoS for the purpose of **Availability**
- Root Cause Analysis (RCA) & troubleshooting
- Sizing machines for migrations e.g. P2V & V2V

Performance bottleneck troubleshooting

What if the service is up but does not perform well?...

Namely, users and customers are complaning about latencies are are saying "it is slow".

==> need to find the performance bottleneck

- System level
- (Database level e.g. MariaDB Slow Query Logs)
- (Application level)

RESOURCE TYPES TO TRACK

- CPU (usage vs. load queue)
- RAM USAGE (& RAM BUS)
- DISK I/O
- NETWORK TX/RX PER INTERFACE

Sizing migrations

Know what resources you need

- for P2V & V2V
- for P2C & V2C (cloud migrations)

Note another way to go is to give max power to all guests and closely monitor their consumption (private cloud only)

Tools for sizing: Zabbix API

Tools for troubleshooting

as for deep-dive troubleshooting

how to check for CPU usage and load queue manually?

==> CPU

```
uptime
top -b
htop
```

X11

```
xload
conky
gkrellm
```

XEN

```
xentop -b -i 1
#--> CPU(sec) CPU(%)
```

How to check for RAM usage manually?

==> RAM

look at the last col (w/o buffer/cache)

```
free -m
```

```
htop
```

XEN

```
xentop -b -i 1
```

```
--> MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%)
```

How to check for DISK I/O manually?

==> DISK I/O

live disk i/o (disable SAR)

```
apt install sysstat
ls -lF /etc/cron.d/sysstat
ls -lF /etc/cron.daily/sysstat
vi /etc/default/sysstat

ENABLED="false"

iostat -d 30 /dev/sda
iostat -x /dev/sda #--> %util
```

like `top` but for disk i/o

```
apt install iotop
iotop -b -n 1
```

XEN

```
xentop -b -i 1
#--> VBD_RD     VBD_WR
```

How to check for NETWORK INTERFACE TX/RX manually?

==> NETWORK TX/RX PER INTERFACE

```
iftop -i eth0
iptraf
trafshow
nload
nethogs eth0
vnstat -i eth0
```

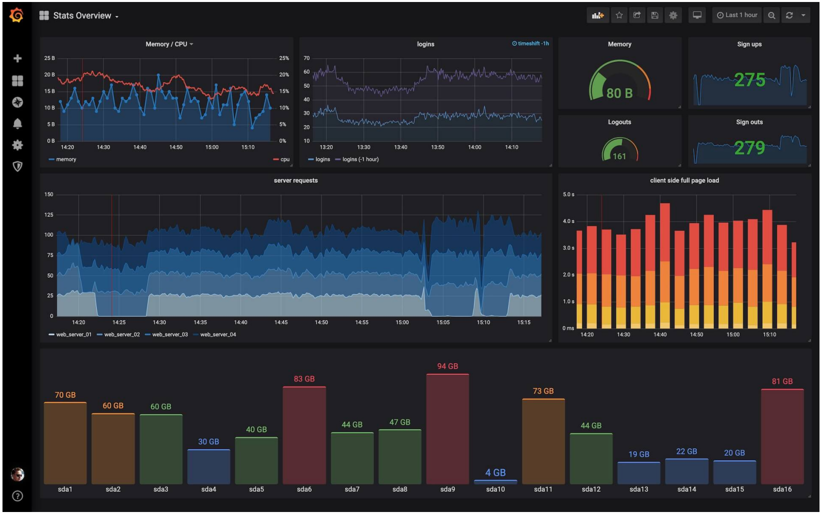
XEN

```
xentop -b -i 1
#--> NETTX(k) NETRX(k)
```

And many others…

```
bmon, bwm-ng, cbm, slurm, tcptrack, netload, collectl, speedometer,
    pktstat, netdiag/netwatch, ifstat, dstat
```

Performance graphs



…that was just some UI (Grafana)

Performance graphs

Goals per system

- See how well your bare-metal systems are sized
- *idem* for guests

Spot the waste (and possibly a DoS attack) e.g.

- Who's using 100% ram?
- Who's using 100% disk i/o?

Goals per hypervisor

- See how well your cluster farm is behaving
- (is the orchestrator doing its job?)
- RAM over-commitment vs. TMEM
- –> 70-90% is good (depending on your cluster size)

and beyond the 4 resource types

- Virtual disks' thin-provisioning

Various ways to get the metrics

- Agents (auto-deploy)
- Hypervisors
  - XEN `xentop`
  - XEN light library
  - some KVM equivalent? // LAB
  - possible from VMware ESXi or vSphere? // LAB
- SNMP

App & services' QA

- `ping` ***response time***

Business logic monitoring

Activities e.g.

- How many connections…
- How many users…
- How many purchases & ratio…

Metric exporters / collectors / scrappers / forwarders

- pull (prometheus scrape) vs. push (influxdb)
- exporter & scrape vs. direct send…

tsdb / dashboards / collectors

- Prometheus / VMetrics
- Prometheus / VMetrics built-in, Grafana
- * ==> Prometheus scrape, Fluent-Bit

tsdb / dashboards / collectors

- InfluxDB
- InfluxDB built-in, Grafana
- * ==> Telegraf, Glances

tsdb / dashboards / collectors

- Graphite/Carbon
- Graphite/Carbon built-in, Grafana
- Statsd

tsdb / dashboards / collectors

- ELK v8 (time-series feature)
- ?
- ?

tsdb / dashboards / collectors

- M/Monit
- M/Monit built-in
- Monit

tsdb / dashboards / collectors

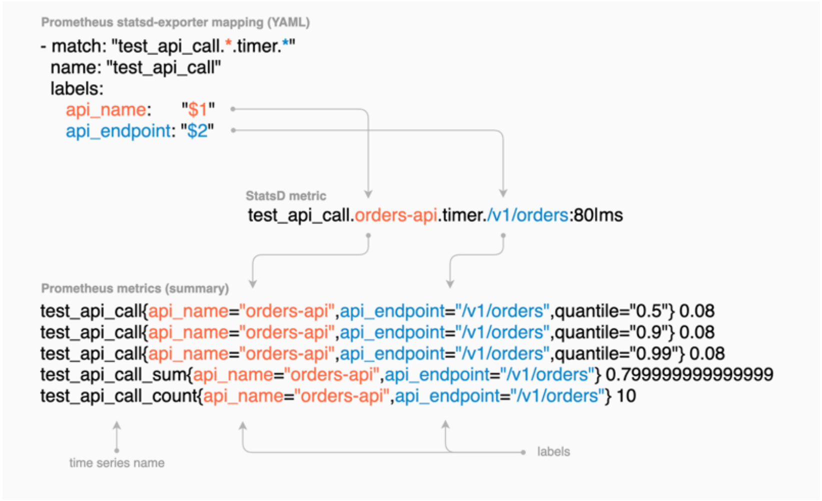
- Zabbix - which underlying TSDB?
- Zabbix built-in
- Zabbit agent, remote check

tsdb / dashboards / collectors

- RRD
- MRTG, RRDtool, Cacti
- Collectd

DIY dashboards

- Highcharts/Highstock - wants json & displays charts live
- Spark - text-based utf-8 bars


Prometheus statsd-exporter mapping (YAML)
StatsD metric
Prometheus metrics (summary)
time series name
labels

```
- match: "test_api_call.*.timer.*"
  name: "test_api_call"
  labels:
    api_name:     "$1"
    api_endpoint: "$2"
```

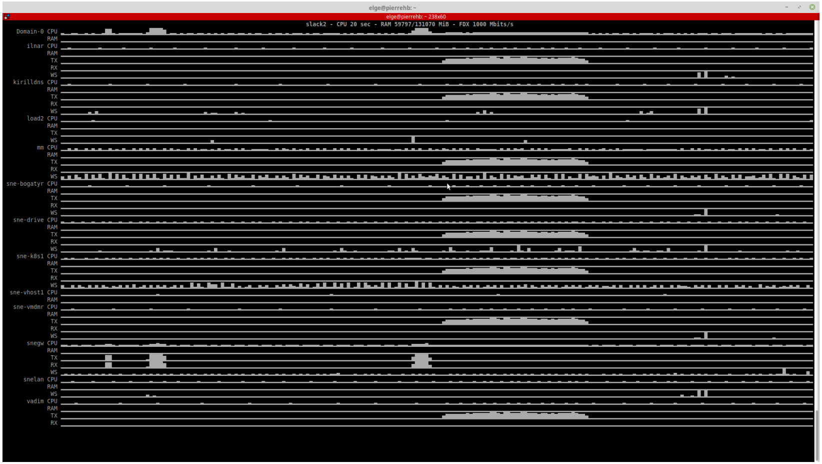
```
test_api_call.orders-api.timer./v1/orders:80|ms
```

```
test_api_call{api_name="orders-api",api_endpoint="/v1/orders",quantile="0.5"} 0.08
test_api_call{api_name="orders-api",api_endpoint="/v1/orders",quantile="0.9"} 0.08
test_api_call{api_name="orders-api",api_endpoint="/v1/orders",quantile="0.99"} 0.08
test_api_call_sum{api_name="orders-api",api_endpoint="/v1/orders"} 0.7999999999999999
test_api_call_count{api_name="orders-api",api_endpoint="/v1/orders"} 10
```

statsd to prometheus format - source: dev.to/kirklewis

Protocols

- Prometheus - TCP HTTP(S) - OpenMetrics
- Statsd - KISS and ultra-light UDP
- Telegraf - ?



```
elge@pierrehb: ~
elge@pierrehb: ~ 238x60
                    slack2 - CPU 20 sec - RAM 59797/131070 MiB - FDX 1000 Mbits/s
  Domain-0 CPU
           RAM
     ilnar CPU
           RAM
            TX
            RX
            WS
 kirilldns CPU
           RAM
            TX
            RX
            WS
     load2 CPU
           RAM
            TX
            WS
        mm CPU
           RAM
            TX
            RX
            WS
sne-bogatyr CPU
           RAM
            TX
            RX
            WS
 sne-drive CPU
           RAM
            TX
            RX
            WS
  sne-k8s1 CPU
           RAM
            TX
            RX
            WS
sne-vhost1 CPU
           RAM
 sne-vmdmr CPU
           RAM
            TX
            RX
            WS
     snegw CPU
           RAM
            TX
            RX
            WS
    snelan CPU
           RAM
            WS
     vadim CPU
           RAM
            TX
            RX
```

No Web Required

LOAD TEST ACCEPTANCE

How to benchmark vs. stress-test?…

==> Benchmarking == dedicated resources (ideally bare-metal)

==> Stress-test == just push-up the volume

LOAD STRESS CPU

assuming 16 cores

```
stress --cpu 16
openssl speed -multi 16
```

LOAD STRESS RAM

```
stress --vm 16 --vm-keep
```

alternative to avoid OOM

```
mkdir -p ram/
mount -t tmpfs -o size=7168M tmpfs ram/
dd if=/dev/zero of=ram/ramload bs=1M
```

LOAD STRESS DISK

Get some idea about disk's speed

```
hdparm -tT /dev/sda
```

Stress the disk

```
time dd if=/dev/zero of=device-or-file bs=1G count=30
bonnie++ ...
stress --io 16
```

LOAD STRESS NETWORK INTERFACES

Flood the network in one direction (UDP)

```
iperf3 -c -u x.x.x.x
```

–or– regulate while checking how much packets got there (TCP)

```
iperf3 -c x.x.x.x
```

MOAR

<https://github.com/akopytov/sysbench>

–> for both system and databases

// Questions on performance monitoring?

Network diagrams

Online tools

- draw.io
- ASCIIFlow
- ASCIIFlow (legacy)

Desktop

- MS visio (proprietary)
- Dia

On-premises

- ASCIIFlow on-premises[1] and without OAuth2

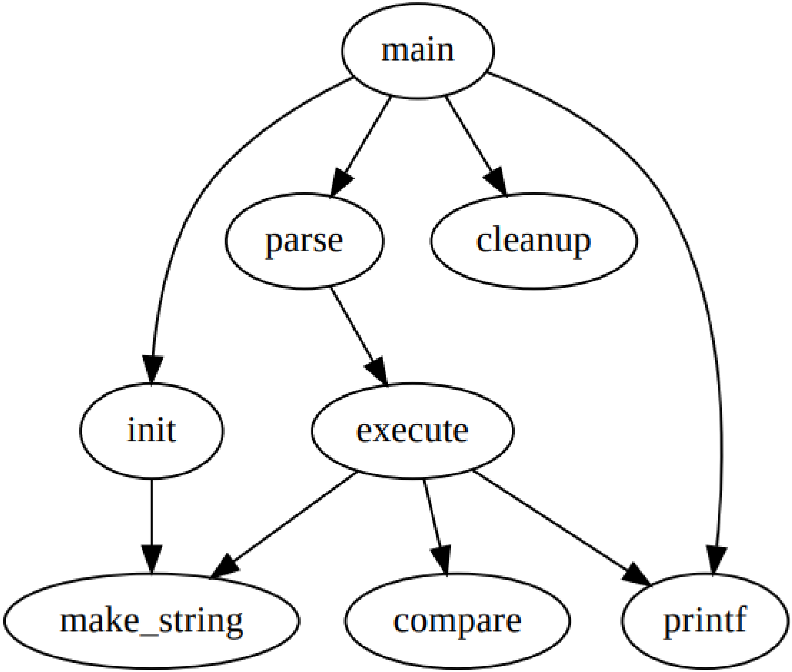
CLI

- ~Graphviz

[1] lewish / asciiflow, <https://github.com/lewish/asciiflow>

Graphviz flow graph example

```
digraph G {
    main -> parse -> execute;
    main -> init;
    main -> cleanup;
    execute -> make_string;
    execute -> printf
    init -> make_string;
    main -> printf;
    execute -> compare;
}
```


main
parse
cleanup
init
execute
make_string
compare
printf

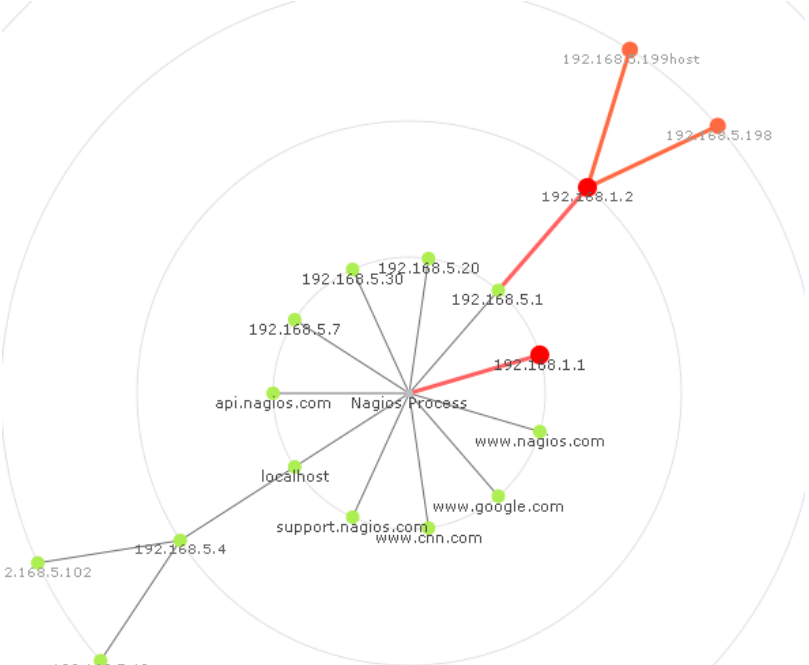
Network Management System (NMS)

What is the difference between NMS and monitoring server?...

==> includes network-oriented and specific features

- all SNMP (no agents)
- network discovery & maps
- network **health** maps
- network performance graphs

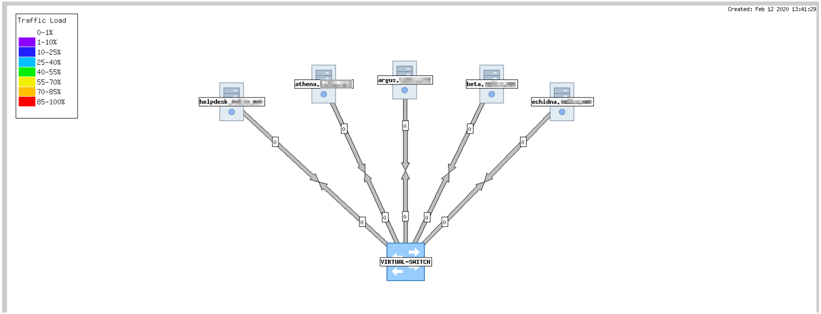
Hypermap plug-in for Nagios



Network Weather Map (standalone?)

OVHcloud Network Weathermap
<http://weathermap.ovh.net/>

Network Weather Map // librenms.org



networkmap.js

Scaleway Netmap
<http://netmap.scaleway.com/>

// any questions on NMSen?

Servers & UIs vs. Agents

store data and visualize

Need to differenciate:

- agents
- metrics collectors
- servers (listeners)
- storage
- UIs

Any monitoring products in mind?…

can-do-it-all - FOSS

- Nagios Core w/ performance data plug-in
- **Zabbix –** server & UI
  - (all kinds of tests)
- Munin
  -
    - RRDtool

performance data only (no alerts) – FOSS

- **InfluxDB –** server & UI
  - - Telegraf (recommended)
  - - Glances (experimental)

UI only (needs TSDB storage) – FOSS

- Grafana frontend

The Nagios situation

- Nagios XI incl. performance graphs (are some parts closed-source?)

Nagios forks

- Centreon
- Icinga 2
- CheckMK

The Monit situation

FOSS

- Monit – agent
  - sends alerts on its own
  - collects and sends data to M/Monit
- Monit Graph – server & UI

Proprietary

- M/Monit – server & UI

unsorted – FOSS

- Pandora FMS?
- Sentry?

mixed FOSS & proprietary?

- Datadog (only agent is open-source?)

proprietary

- Solarwinds Server Application Monitor – *major leakage lately…*
- Paessler PRTG
- M/Monit (server & UI only)

specific features here and there

Android & iPhone app

- **Zabbix**
- LibreNMS
- …?

Network Health Maps

- Network Weather Map (PHP) – compatible e.g. w/ LibreNMS & Cacti
- networkmap.js
- netTransformer

Note it's also possible to do all that with originally system-oriented monitoring engines

- Nagios Core – plug-in Hypermap
- Nagios XI – is there something already?
- **Zabbix** – got Maps feature

// any questions on those various kinds of products?

Log servers

one more screen in operations (NOC) room

- alerts on incident monitoring
- graphs on performance monitoring
- alerts on **errors and unusual logs**
- alerts on network anomalies (IDS heuristics)

Two ways to consider log centralization

- send everything and face the load (expect delays…)
- send only warnings and errors and keep it as clean and empty as possible

Besides, the latter would make appropriate for an in-memory database…

Casual and old-school setup example

on the server

```
syslogd # without -s
```

on the client

```
vi /etc/syslog.conf

*.warn      @log-server
```

Log server products

FOSS

- Graylog – *with UI*
- ELK – *with UI*
- sysklogd / rsyslog / syslog-ng / bsd syslogd – *text-mode dashboard*
- DIY logstash –> MongoDB or Redis

Commercial

- Splunk
- Loggly
- LogZilla (apparently only some modules are FOSS)

Log server features

- easy access to dev groups (although `group:adm` is alright)
- can trigger alerts when abnormal logs are spotted
- possibly driven by heuristics (what AI plug-in for graylog?)

Beware it takes a lot of space, and server load. You might consider sending only application errors to the log farm.

```
nginx error logs
apache error logs
php-fpm logs (that's only errors anyhow)
```

It might also be interesting to send system errors, but you need to tune priorities for that.

Value	Severity	Description
0	emergency	System is unusable
1	alert	Action must be taken immediately
2	critical	Critical conditions
3	error	Error conditions
4	warning	Warning conditions
5	notice	Normal but significant conditions
6	info	Informational messages
7	debug	Debug-level messages

What kind of agents do we need to send the logs there?…

Log grabbers

```
syslog UDP                                --> standard syslog
logstash                                  --> ELK
collector-sidecar (filebeat / nxlog)      --> Graylog
```

Note there’s a way to setup logstash to send to MongoDB or Redis directly

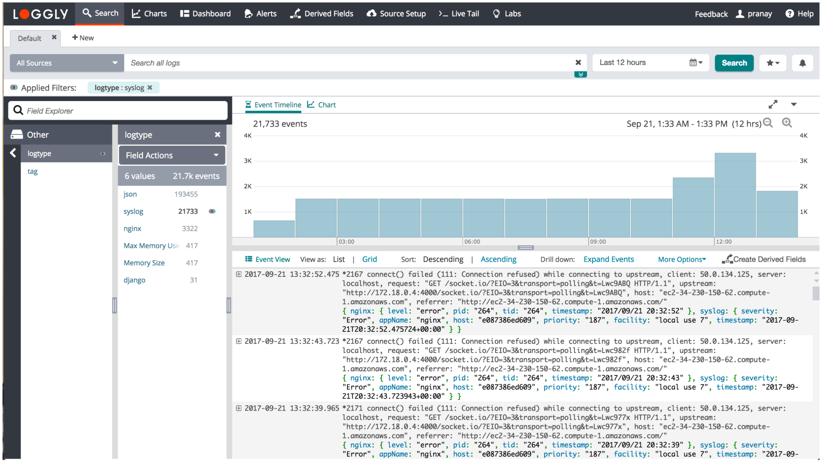
Storage for logs

(not just time-series metrics for once)

`Graylog storage`	`ElasticSearch & MongoDB`
`ELK storage`	`ElasticSearch (& MongoDB?)`
`phpsyslog-ng`	`MySQL`

ok so far, we got the logs centralized, but what now?…

==> *we need to visualize the logs with specific search patterns*



```
⊞ 2017-09-21 13:32:52.475 *2167 connect() failed (111: Connection refused) while connecting to upstream, client: 50.0.134.125, server:
                          localhost, request: "GET /socket.io/?EIO=3&transport=polling&t=Lwc9ABQ HTTP/1.1", upstream:
                          "http://172.18.0.4:4000/socket.io/?EIO=3&transport=polling&t=Lwc9ABQ", host: "ec2-34-230-150-62.compute-
                          1.amazonaws.com", referrer: "http://ec2-34-230-150-62.compute-1.amazonaws.com/"
                          { nginx: { level: "error", pid: "264", tid: "264", timestamp: "2017/09/21 20:32:52" }, syslog: { severity:
                          "Error", appName: "nginx", host: "e087386ed609", priority: "187", facility: "local use 7", timestamp: "2017-09-
                          21T20:32:52.475724+00:00" } }
⊞ 2017-09-21 13:32:43.723 *2167 connect() failed (111: Connection refused) while connecting to upstream, client: 50.0.134.125, server:
                          localhost, request: "GET /socket.io/?EIO=3&transport=polling&t=Lwc982f HTTP/1.1", upstream:
                          "http://172.18.0.4:4000/socket.io/?EIO=3&transport=polling&t=Lwc982f", host: "ec2-34-230-150-62.compute-
                          1.amazonaws.com", referrer: "http://ec2-34-230-150-62.compute-1.amazonaws.com/"
                          { nginx: { level: "error", pid: "264", tid: "264", timestamp: "2017/09/21 20:32:43" }, syslog: { severity:
                          "Error", appName: "nginx", host: "e087386ed609", priority: "187", facility: "local use 7", timestamp: "2017-09-
                          21T20:32:43.723943+00:00" } }
⊞ 2017-09-21 13:32:39.965 *2171 connect() failed (111: Connection refused) while connecting to upstream, client: 50.0.134.125, server:
                          localhost, request: "GET /socket.io/?EIO=3&transport=polling&t=Lwc977x HTTP/1.1", upstream:
                          "http://172.18.0.4:4000/socket.io/?EIO=3&transport=polling&t=Lwc977x", host: "ec2-34-230-150-62.compute-
                          1.amazonaws.com", referrer: "http://ec2-34-230-150-62.compute-1.amazonaws.com/"
                          { nginx: { level: "error", pid: "264", tid: "264", timestamp: "2017/09/21 20:32:39" }, syslog: { severity:
                          "Error", appName: "nginx", host: "e087386ed609", priority: "187", facility: "local use 7", timestamp: "2017-09-
```

source: loggly.com

any fancy log server products products in mind?…

==>

FOSS

- phpsyslog-ng
- ELK
- Graylog
- LogZilla

Proprietary

- Splunk
- Loggly (interface based on phpsyslog-ng?)
- …?

ideally make use of both system logs + send logs to log server

- make it possible for sysadmins and devs to troubleshoot on host
- SSHGuard won’t work here unless you double the logs (local + remote)

Avoid flood attacks on storage

This is why it is recommended to separate those

```
/
/home
/var
/tmp
```

- can prevent users and admins to log in…

usually not an issue anymore, thanks to storage capabilities and default log retention settings

Note: do not forget to setup `logrotate` *or* `newsyslog` *if you deployed your own custom build*

Also in conjuction with incident monitoring, you might

good practice? keep error logs empty

// any questions on log servers?

This is the end