The challenge of external data

Martin Simons
6 februari 2024

Whoami?

Born a Millenium ago

Unofficial husband

Official father of three daughters

Love to drink beer

Debian

Get it right kind of guy

Self employed since 1998

Hit me on LinkedIn, Webhuis

Old Configuration Management

It all started in the nineties...

With bare metal machines

Having many combined roles on one machine

An ntp server was stuffed to a central machine:

– The name server

Bigger data centres to meet the Internet needs

The config management problem was born

New Configuration Management

Virtual Machines emerged

Containers

IOT

Small units having a simple role

There are **behavioral** aspects to actors in the field

Throwing everything away

So we do config management for 30+ years

- Tons of scripts have emerged

- Numerous tools come and go

- Where has all the information gone?

- Why has all the information gone?

There is high value in the information

Data / Information is more **persistent** than tools

A fresh start every day

It is natural and very human

There never is time to do it right,

But there is always time to do it again

Orchestration, The Holy Grail

Aspects of Orchestration

- There is a **plan**
- There is a **Desired State**

Orchestration

The conductor

- Has a Desired State in mind

- Conveys the Desired State he desires

- The different actors focus on their own role

- Actors do not mind other roles too much

- Works highly **imparative**

- Might even **enforce** synchronisation of moves

- Actors are highly dependend

- Is continuously sending messages

- Does he arrive at the **Desired State**

Orchestration, Span of Control

Limits of Orchestration

Trying to go beyond the limits is looking **funny**

- When do we reach the Desired State?

- To what extend do we have a desired state?

- Is it possible to arrive at the desired state?

- Is the Desired State persistent?

- No autonomy in the above examples

- Highly centralised

- Hardly any interaction between the actors

Orchestration?

Autonomous Actors – 1

Real World

- Actors and the Desired State(s)
- Can will still speak of a Desired State?
- How do we arrive at the desired state?
- Is the Desired State persistent?
- Enforcing becomes very difficult
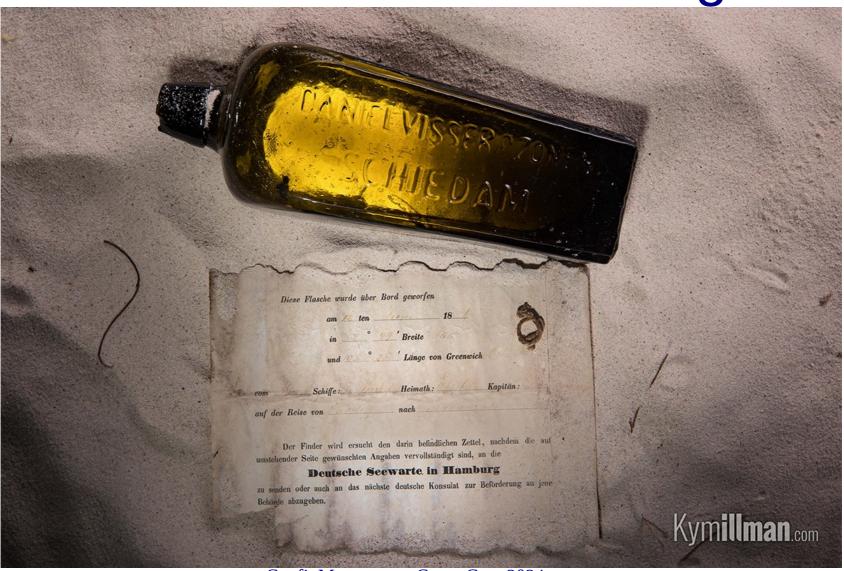- Autonomy vs dependency

Autonomous Actors – 2

Real World

- It looks messy, but it works if….
- Actors use a common infrastructure
- Actors go from a(n) to b(m)
- Actors are aware of other Actors
- Actors use standardised messages to interact
- **Intentions** are being **advertised**

Messaging

All sorts of messages involved

- Interaction

- Behavior

- Types of messages

- Context of messages

- Ways of managing messages

Classic broadcast message

Broadcasting aspects

- One way delivery

- TV Radio

- Flyering

- Message in a bottle

- Verified delivery

- Message brokering

Formal messaging system

Postal services aspects

It was the only way

- Upstream

- Processing

- Downstream

- The Postal service acts like a broker

- Single point of failure, strike

There are ways of signaling

Most of them broadcasting

- Lighthouse

- Telephone tree

- Smoke signals

Messaging in an IT landscape

Why would there be messages?

- The IT World gets more and more **dynamic**

- Numerous **different** types of devices

- Devices become more and more **intelligent**

- Devices use actual information in their **role**

- Devices need to adapt to change

- Many Devices need **dynamic configuration**

Dealing with information

Every Actor in the field provides information

- Processing information
- Generated **Role** based information
- Valuable for collection and distribution
- Actual and historic information
- Distributing sensitive data out of a database

Enter **Data**

How does Data work?

Data is an App and a PostgreSQL database.

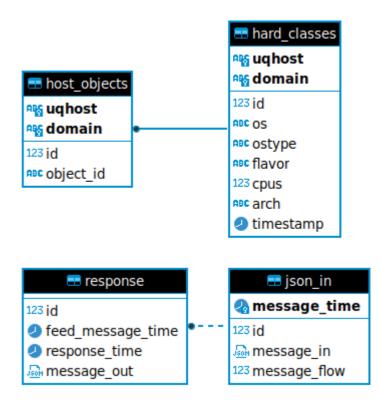The database has three schema's:

- Feeds
- Context
- Knowledge

The App is an object oriented Python3 program

- Three tier (I hope)
- Multi threading
- https://github.com/Webhuis/Data/tree/master
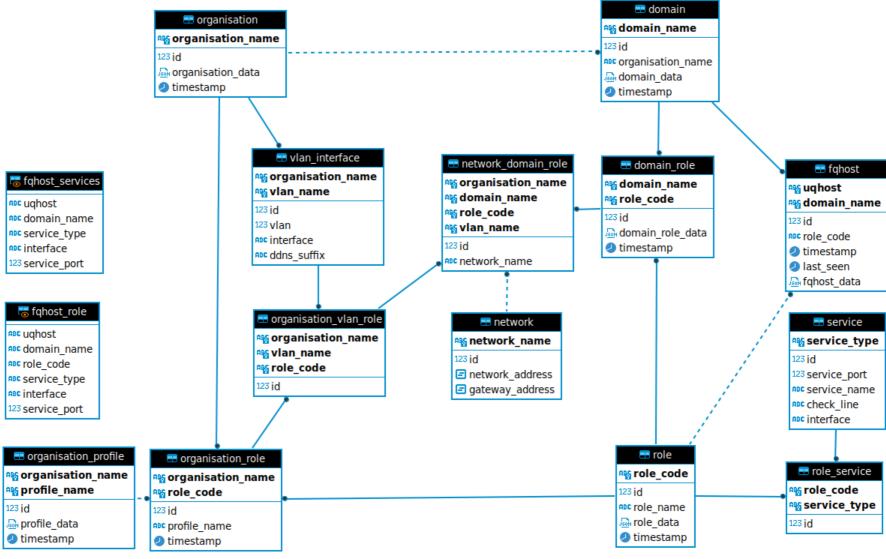
The nosql feed schema

Upstream schema feeds

Making data available

So how becomes data made manageble?

- Make individual nodes **share** their data

- Every message is a (json) container

- Nodes send **feeds**

- Data processes feeds

- Nodes send queries

- Data gives taylored **responses** to queries

The enriched data ER model

How does the complex work?

Data and the Actors exchange messages

- Using 0mq, zeromq or ZMQ

- Periodically

- Complex messages, having containers

- Using the JSON format

- Works for **every** 0mq and JSON **enabled** Actor

- May contain sensitive information

The communication is **not** yet encrypted

TCP/IP supports all

The protocol allows

- Broadcasting

- Veryfied delivery

- Brokered messaging

- **Brokerless** point to point messaging

Every Actor feeds Data

This is the hard_classes feed:

{"message": "feed",

"query": "hard_classes",

"content": {

"feeding_host": "mnmutl0001",

"domain": "webhuis.nl",

"os": "linux",

"ostype": "linux_x86_64",

"flavor": "debian_8",
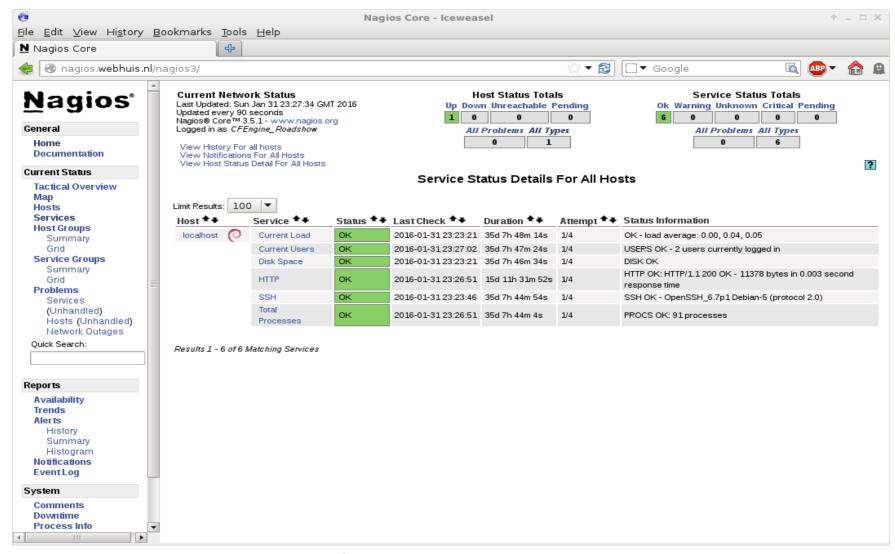
"cpus": "1",

"arch": "x86_64" } }

It is fast!

Message in: 2024-02-05 15:07:47.493

Response out: 2024-02-05 15:07:47.507

- A grand total of **14 msec** (on the Data box)

- Carried out seven selects

- Did at least two inserts

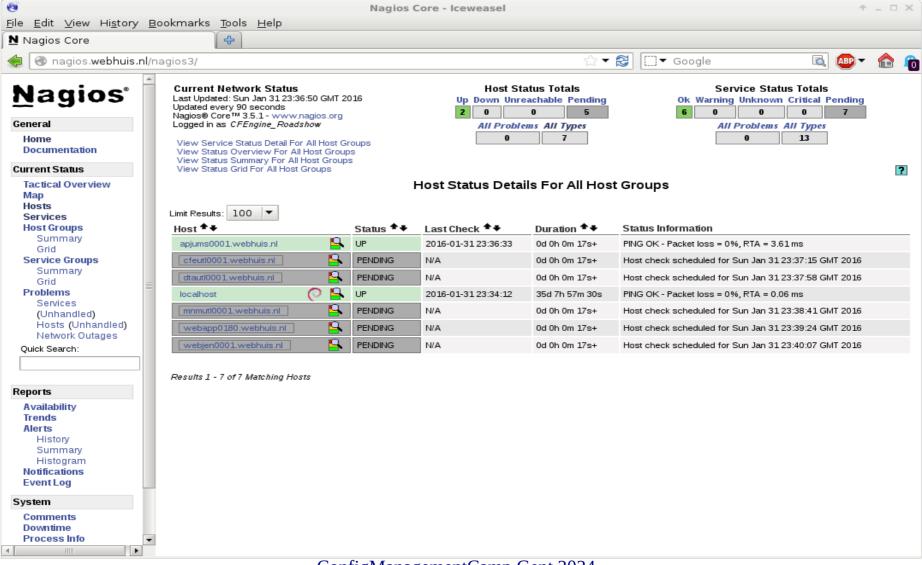- The response size is around 1.5 Kb

Example Nagios Monitor

Actors feed their details

It is a dynamic setup

- Every time an actor comes by

- Data registers details

- The monitor gets the relevant details
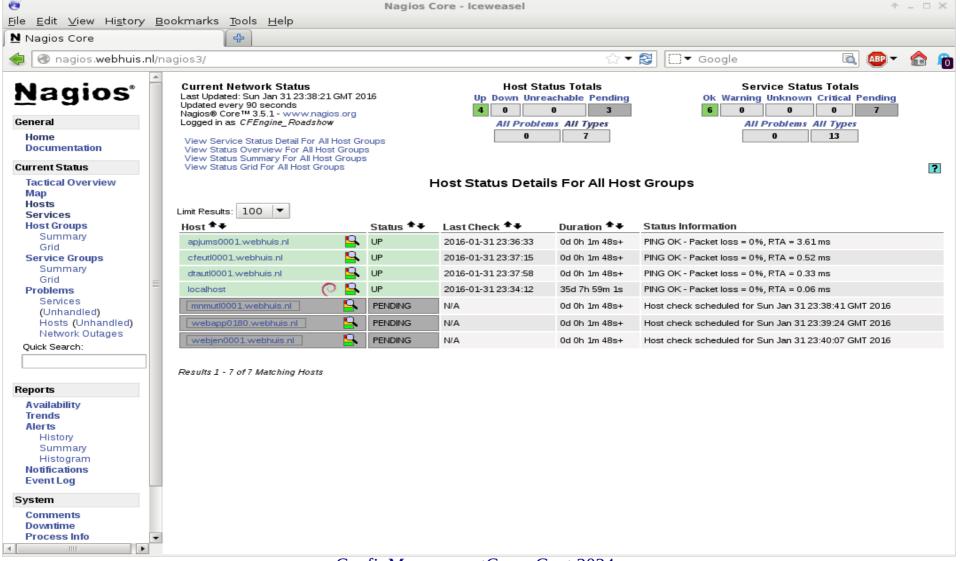
- The Actors **autonomously** process messages
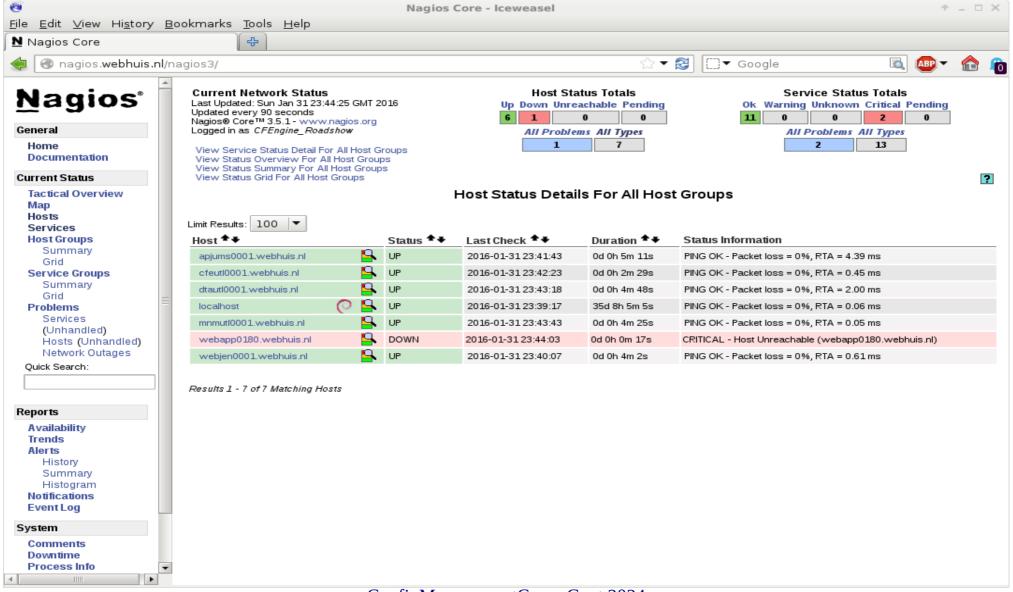
Hosts are added

Response from Data

{"pdnrpdcsw001.sw.webhuis.nl": [{"fqhost_data": ""}, {"Webhuis": [{"organisation_data": [{"dhcp_ldap": "mihrwhpdcsw1.sw.webhuis.nl", "administrator": "martin@webhuis.nl", "ldap_frontend": "ldap.sw.webhuis.nl"}, {"ldap_m": [["100", "ldappdcsw001.sw.webhuis.nl"], ["101", "ldappdcsw002.sw.webhuis.nl"]]}, {"primary_dns": [{"mdns": "powerdns.webhuis.nl"}, {"mdns_key": "/etc/dhcp/webhuis.nl.update"}]}, {"local_networks": ["10.0.0.0/8", "192.168.0.0/16", "213.127.130.120/29"]}, {"ttl": "300"}, {"ntp_server": "ntp.time.nl"}]}]}, {"domain_data": [{"ldap_base": "dc=sw,dc=webhuis,dc=nl"}, {"local_resolvers": ["10.68.171.24", "10.10.117.24", "10.10.217.24"]}, {"searchlist": "sw.webhuis.nl"}]}, {"role_data": ""}, {"domain_role_data": ""}, {"domain_role_network": [[{"vlan_name": "admin", "vlan": 22, "interface": "eth2", "ddns_suffix": "-ssh", "network_name": "admin_sw", "network_address": "10.168.71.0/24", "gateway_address": "10.168.71.0"}], [{"vlan_name": "support", "vlan": 123, "interface": "eth1", "ddns_suffix": "", "network_name": "support_sw", "network_address": "10.68.171.0/24", "gateway_address": "10.68.171.0"}]]}, {"organisation_profile": [["Webhuis", "pdnr", "std_sup", {"organisation_profile": [{"etc_hosts_nic": "support"}, {"admin_nic": "admin"}, {"nic_trick": ["0", "1"]}]}]]}, {"pdnr": [[{"service_port": 22, "service_name": "sshd", "check_line": "check_tcp\\!22", "interface": "eth2"}], [{"service_port": 53, "service_name": "Powerdns server", "check_line": "check_tcp\\!53", "interface": "eth1"}]]}]}

Services of hosts join in

Do I smell a desired state somewhere?

Convergence

Actors

- Will exchange information periodically
- **Convergently** work towards the Desired state

Message timeouts

The duration of a message exchange

- The response times are normally distributed

- Timeout .05 seconds, on a local connection

- Important to configure to prevent overload

Use cases

When an Actor needs actual information

- Monitoring servers in a dynamic landscape

- Certificate distribution

- About other Actors that are around

Use cases

When you need information

- Performance analysis

- Capacity analysis

- Historic information

- Big Data analysis

Conlusion?

Keep the gold to yourself,

Stop throwing it away

Discussion