FIRST IMPRESSIONS OF SALTSTACK AND RECLASS

DENNIS VAN DOK

HEPIX SPRING 2018 WORKSHOP – MADISON, WI, THURSDAY 2018-05-17

A NEW CONFIGURATION MANAGEMENT SYSTEM?

We've been using Quattor since the early DataGrid days.

Changing landscape; grid services see less innovation, new CM systems emerged along with growing cloud deployments.

If there ever was a moment to do it, this was it!

ABOUT THIS TALK

- not a technical talk
- the journey is more interesting than the destination
- we're got plenty of the road ahead of us

A NEW SYSTEM!

Credits to Andrew Pickford!

Looked at quattor upgrade:

- a lot of work
- smallness of quattor community
  - they certainly wanted to help
  - not easy to get going based on available documentation

CONSIDERING SEVERAL ALTERNATIVES

(But some were rejected outright based on personal prejudice.)

An honest comparison would have been too much work.

Two candidates came very close: **Saltstack** and **Ansible** with no obvious winner.

Saltstack came out ahead by a nose on technicalities.

(Ansible would have served us just fine.)

WHAT WE LIKED

(Based on previous experiences)

- we really liked the state concept of Saltstack (similar to Quattor).
- Everything is YAML and Python. (And, ok, Jinja2.)
- Nice integration with Reclass (more later).
- Test mode shows what *would* change.

A FIRST LOOK AT SALTSTACK

Discussed (a bit) at HEPiX before.

- 2016, Sandy Philpott, Site report,
  https://indico.cern.ch/event/531810/contributions/2314173/
- 2017, Owen Synge, Technical talk,
  https://indico.cern.ch/event/595396/contributions/2544138/

Widely used in various open source communities.

THIS IS NOT A TECHNICAL TALK

(But anyway…)

- master/minion system
- minions controlled by defined **states**
- static data provided by **pillars**
- states are logically bundled by **formulas**
- states are implicitly **ordered** by dependencies

WHAT GOES WHERE

data source	kind of data	typical examples
pillar	static per-node	server name, ip address
formula	states related to a single aspect	mysql, iptables
state	elementary settings	installed packages, running services

EXAMPLE OF STATE RUN IN TEST MODE

```
      ID: /etc/nova/nova.conf
Function: file.managed
  Result: None
 Comment: The file /etc/nova/nova.conf is set to be changed
 Started: 15:37:01.083553
Duration: 380.062 ms
 Changes:
          ----------
          diff:
              ---
              +++
              @@ -158,6 +158,7 @@
               # * ``hyperv.HyperVDriver``
               #  (string value)
               #compute_driver=<None>
              +compute_driver=libvirt.LibvirtDriver

               #
               # Allow destination machine to match source for resize. Useful when
```

ORGANISING OUR DATA WITH RECLASS

We separated the

- moving parts (*states*) that are the same for all our nodes from the
- static data specific to each node (pillar).

The pillar is provided by Reclass.

RECLASS

A recursive classifier, collecting static hierarchical information about nodes providing *pillar* data.

Originally http://reclass.pantsfullofunix.net/, but the most active fork at the moment is https://github.com/salt-formulas/reclass/. Our version currently is https://github.com/AndrewPickford/reclass/.

RECLASS IN A NUTSHELL

(Remember, not a technical talk!)

- Each node specifies which *classes* it belongs to;
- each class is a file in a hierarchy (i.e. directory structure);
- each class file lists more classes and/or parameters;
- later classes override (simple values) or merge (lists) values from earlier classes.

RECLASS EXAMPLE

Example, slightly simplified. This is a dCache master node in our testbed.

```
classes:
  - cluster.ndpf.testbed.dcache
  - hardware.vm.xen.standard
  - os.linux.redhat.centos.7
  - role.server.dcache.plain.master
environment: pre-prod
parameters:
  _hardware_: (here be the VM provisioning parameters)
```

Nikhef

here is cluster/ndpf/testbed/dcache/init.yml:

```
classes:
  - cluster.ndpf.testbed
parameters:
  _cluster_:
    name: dcache testbed
    dcache_version: 3.1
    dcache_carbon_server: ${_cluster_:monitoring_satellite}
    dcache_nfs_allowed_ipv4:
      - ${_site_:networks:ipv4:stbcnet}
      - ${_site_:networks:ipv4:wnnet}
```

cluster/ndpf/testbed/init.yml:

```
classes:
  - cluster.ndpf
parameters:
  _cluster_:
    name: testbed
    monitoring_satellite: vaars-03.nikhef.nl
```

Note that `_cluster_:name` is given here, but the class cluster.ndpf.testbed.dcache overrides it.

WHAT DATA GOES WHERE

- Reclass allows more freedom in layout of data
- Following a logical structure rather than what is imposed by a system
- Only simple constructs allowed; complicated programming relegated to states

SHORTCOMINGS

Reclass is not without its shortcomings. It needed work to make it do what we wanted, and was (therefore) almost rejected.

We still went ahead and fixed it.

REDEEMING QUALITIES

Written in python which is nice and forgiving to programmers.

Our patches are available on Github, and we're looking to integrate with versions maintained by the salt-formulas people.

Nikhef

ADDED FEATURES

Exports

allow extraction of info from other nodes. This is conceptually related to the *salt mine* but comes in at an earlier stage of the processing chain.

References

were enhanced to allow nesting; overriding values will do merge instead of replace when values are lists or dicts.

Git backend

works just like the git backend for Salt, so data is taken straight from a repository/branch.

IMPROVED ERROR HANDLING AND REPORTING.

```
      - Failed to load ext_pillar reclass: ext_pillar.reclass: →
…-> cc2.cloud.ipmi.nikhef.nl
          Cannot resolve ${_cluster_:some:value}, at →
…_cluster_:monitoring_satellite, →
…in yaml_fs:///srv/salt/env/dennisvd/classes/cluster/ndpf/cloud/init.yml
```

FORMULAS

All the moving parts are grouped by formulas.

apache, authconfig, autofs, backupninja, bind, certificates, cinder, cobbler, contrailctl, cups, cvmfs, dcache, dell_mdsm, docker, elasticsearch, eos, galera, git, glance, grafana, graphite, grid, haproxy, hardware, horizon, icinga, iptables, keepalived, kerberos, keystone, kibana, linux, logrotate, logstash, maui, memcached, munge, mysql, neutron, nfs, nikhef, nova, ntp, pacemaker, pakiti, php, postfix, postgresql, prometheus, python, rabbitmq, reclass, repo-mirrors, rsync, rsyslog, salt, sanity-check, secure, tftpd_hpa, torque, zookeeper

PROS AND CONS

Pros:

- encapsulate a functional element
- forms a clear conceptual boundary
- places complexity where we want to handle it

Cons:

- many repositories (requires scripting)
- mixed quality (often only tested on Debian)

SINGLE OR SEPARATE REPOSITORIES?

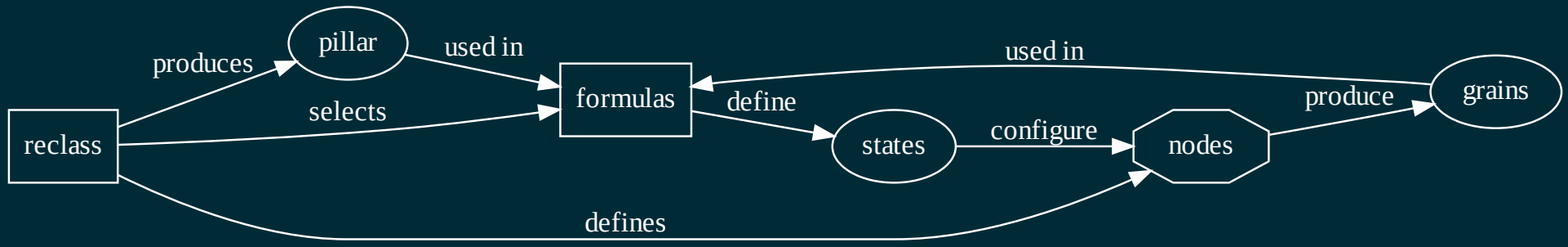
Choice:

- put all formulas in a single repository, or
- keep all formulas in their own repository

FORMULAS AND RECLASS

- Formulas are driven by pillar data
- This makes them integrate well with reclass.

INFORMATION FLOW AND RELATIONSHIPS



Nikhef

VERSION CONTROL

- keep everything in private Gitlab
- master branch in Gitlab **defines** what is in **production**
- other branches correspond to environments

GIT AS A WORKFLOW DRIVER

- git push to master determines what is in production
- manual deploy initiated thereafter still necessary
- we needed a pre-production testbed to test changes before the push
- we needed a way to sync up the many formula repositories

PRE-PRODUCTION

- Each type of system has its counterpart in pre-production.
- Pre-production looks at a local checked out version of the master branch.
- Variants for treating updates:
  - minor changes can be applied and tested before committing
  - major updates are tested in other environments and handled via git merging of branches

PEPPER WRAPPER

High level pepper scripts to replace low level salt.

- dealing with multiple repositories
- test
- deploy
- commit
- other git commands

```
Pepper-deploy
```

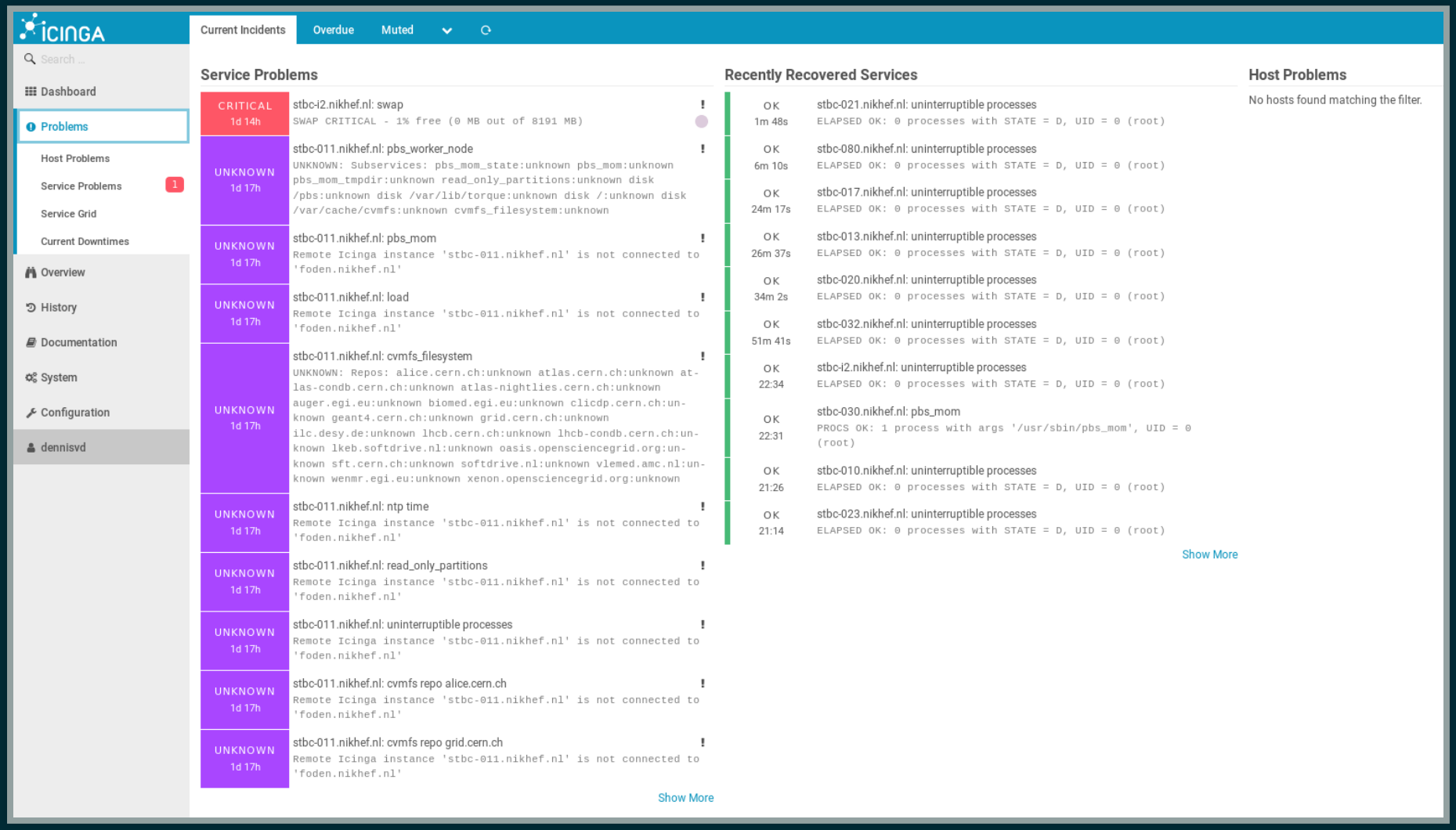
will stagger updates to prevent overload on the master.

ENVIRONMENTS

Environments correspond to branches in git.

- Each newly introduced formula must have branches for every environment.
- Pre-production is the exception, because it looks at the master branch (but actually a local checkout).
- People have their 'own' environment for testing and development purposes.
- possibility to ‘move’ a machine between environments

MONITORING


icinga
Current Incidents
Overdue
Muted
Search ...
Dashboard
Problems
Host Problems
Service Problems 1
Service Grid
Current Downtimes
Overview
History
Documentation
System
Configuration
dennisvd
Service Problems
CRITICAL 1d 14h
stbc-i2.nikhef.nl: swap
SWAP CRITICAL - 1% free (0 MB out of 8191 MB)
UNKNOWN 1d 17h
stbc-011.nikhef.nl: pbs_worker_node
UNKNOWN: Subservices: pbs_mom_state:unknown pbs_mom:unknown pbs_mom_tmpdir:unknown read_only_partitions:unknown disk /pbs:unknown disk /var/lib/torque:unknown disk /:unknown disk /var/cache/cvmfs:unknown cvmfs_filesystem:unknown
stbc-011.nikhef.nl: pbs_mom
Remote Icinga instance 'stbc-011.nikhef.nl' is not connected to 'foden.nikhef.nl'
stbc-011.nikhef.nl: load
Remote Icinga instance 'stbc-011.nikhef.nl' is not connected to 'foden.nikhef.nl'
stbc-011.nikhef.nl: cvmfs_filesystem
UNKNOWN: Repos: alice.cern.ch:unknown atlas.cern.ch:unknown atlas-condb.cern.ch:unknown atlas-nightlies.cern.ch:unknown auger.egi.eu:unknown biomed.egi.eu:unknown clicdp.cern.ch:unknown geant4.cern.ch:unknown grid.cern.ch:unknown ilc.desy.de:unknown lhcb.cern.ch:unknown lhcb-condb.cern.ch:unknown lkeb.softdrive.nl:unknown oasis.opensciencegrid.org:unknown sft.cern.ch:unknown softdrive.nl:unknown vlemed.amc.nl:unknown wenmr.egi.eu:unknown xenon.opensciencegrid.org:unknown
stbc-011.nikhef.nl: ntp time
Remote Icinga instance 'stbc-011.nikhef.nl' is not connected to 'foden.nikhef.nl'
stbc-011.nikhef.nl: read_only_partitions
Remote Icinga instance 'stbc-011.nikhef.nl' is not connected to 'foden.nikhef.nl'
stbc-011.nikhef.nl: uninterruptible processes
Remote Icinga instance 'stbc-011.nikhef.nl' is not connected to 'foden.nikhef.nl'
stbc-011.nikhef.nl: cvmfs repo alice.cern.ch
Remote Icinga instance 'stbc-011.nikhef.nl' is not connected to 'foden.nikhef.nl'
stbc-011.nikhef.nl: cvmfs repo grid.cern.ch
Remote Icinga instance 'stbc-011.nikhef.nl' is not connected to 'foden.nikhef.nl'
Show More
Recently Recovered Services
OK 1m 48s
stbc-021.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
OK 6m 10s
stbc-080.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
OK 24m 17s
stbc-017.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
OK 26m 37s
stbc-013.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
OK 34m 2s
stbc-020.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
OK 51m 41s
stbc-032.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
OK 22:34
stbc-i2.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
OK 22:31
stbc-030.nikhef.nl: pbs_mom
PROCS OK: 1 process with args '/usr/sbin/pbs_mom', UID = 0 (root)
OK 21:26
stbc-010.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
OK 21:14
stbc-023.nikhef.nl: uninterruptible processes
ELAPSED OK: 0 processes with STATE = D, UID = 0 (root)
Show More
Host Problems
No hosts found matching the filter.

- Relies on the exports mechanism discussed earlier
- Nodes specify
  - what type of thing they are, and
  - the kinds of things anyone interested in monitoring should be looking for.

The monitoring system defines how the actual monitoring is done for all of those things. It gets the list of nodes and services from the inventory.

DEPLOYMENT

- cobbler
- based on exports.
- supported by scripts
- hardware description of a node
  - prescriptive for VMs
  - descriptive for actual hardware

The cobbler node has to manage both production and pre-production, and is the 'odd one out' as it has no pre-production counterpart.

REPOSITORIES

The cobbler server also collects mirrors of various repositories for software installation.

- time-based snapshots
- no dependencies on external repositories in production
- support for both apt and yum repos

SYSTEMS SALTIFIED SO FAR

- dcache
- salt master
- cobbler
- torque/maui (local cluster)
- DNS (in high availability setup)
- monitoring (grafana, icinga)
- NFS server
- EOS
- Openstack (still experimentally)
- more to come

CONCLUSIONS

OPEN PROBLEMS

- Running the inventory with 'broken' nodes
- Performance issues with large deployments

FUTURE

- full automated installations
- pre-provisioning keys (salt, ssh, others)
- orchestration
  - stagger kernel updates
- multi-master
- performance issues
  - where does the system spend most of its time?
  - high load on master
  - addressed by batching updates with pepper scripts
  - the monitoring box will go to 500+ states as we add more systems

LESSONS LEARNED

- New system is a lot of work.
- Organisation of data is more important than mechanics.
- Tradeoff between flexibility in prototying and control in production.
- No truly bad choices, but many secondary factors to consider.
- Look at the specific needs of the team; better find a good match than just go with the most popular system.