# An Overview of Aquilon

James Adams

Science & Technology Facilities Council
Rutherford Appleton Laboratory

2014-02-01

# Overview

- About Me
- Some History
- Aquilon
- Example
- Conclusion

# About Me

- Scientific Computing Department
  - 160 Staff - Daresbury and Rutherford Appleton Laboratories
  - Large scale HPC & HTC facilities, data services and infrastructure
  - Petabytes of storage, tens of thousands of cores.
  - Supercomputers at 23 & 283 in Top500 (25 & 69 in Green500)
- Seven years on GridPP Tier 1 centre for Worldwide LHC Computing Grid
  - Distributed computing grid for particle physicists.
  - 150 computing centres in 40 countries.
  - Everything from the hardware to user services.

# How did we get here?

- 1st Generation — CDB
  - Pan code stored in CVS
  - Basic deployment workflow tooling
  - Global locking quickly caused scaling problems
  - Abandoned by the community, still used by CERN for legacy systems
- 2nd Generation — SCDB
  - Pan code stored in Subversion
  - Tagged deployment workflow based on ant and SVN repository hooks
  - Global deploys cause scaling pain

# (S)CDB

- Similar principles
  - Code → Compile → Commit → Deploy → Repeat
- Neither much more than an environment for writing Pan
  - Some layout guidelines
  - Lack of rules for structure of configuration leads to fragmentation, even within sites
- Inputting lots of systems gets boring quickly
  - Users built custom inventory databases
  - Scripting only goes so far
- But powerful enough to be good enough!

# Motivation

- 2007: Morgan Stanley joined community
  - Outgrown existing system
  - Planning to deploy 20,000+ hosts
  - (S)CDB won't scale to this
- Requirements:
  - Global builds not mandatory
  - Large numbers of users with different privileges
    - e.g. front line support staff
    - Routine operations as documented commands
    - Make changes without editing Pan code
  - Ability to branch configuration for development and testing
    - Test changes without committing to a VCS
    - Deploy hosts from branches
  - Provide structure for configuration

# Something entirely new required

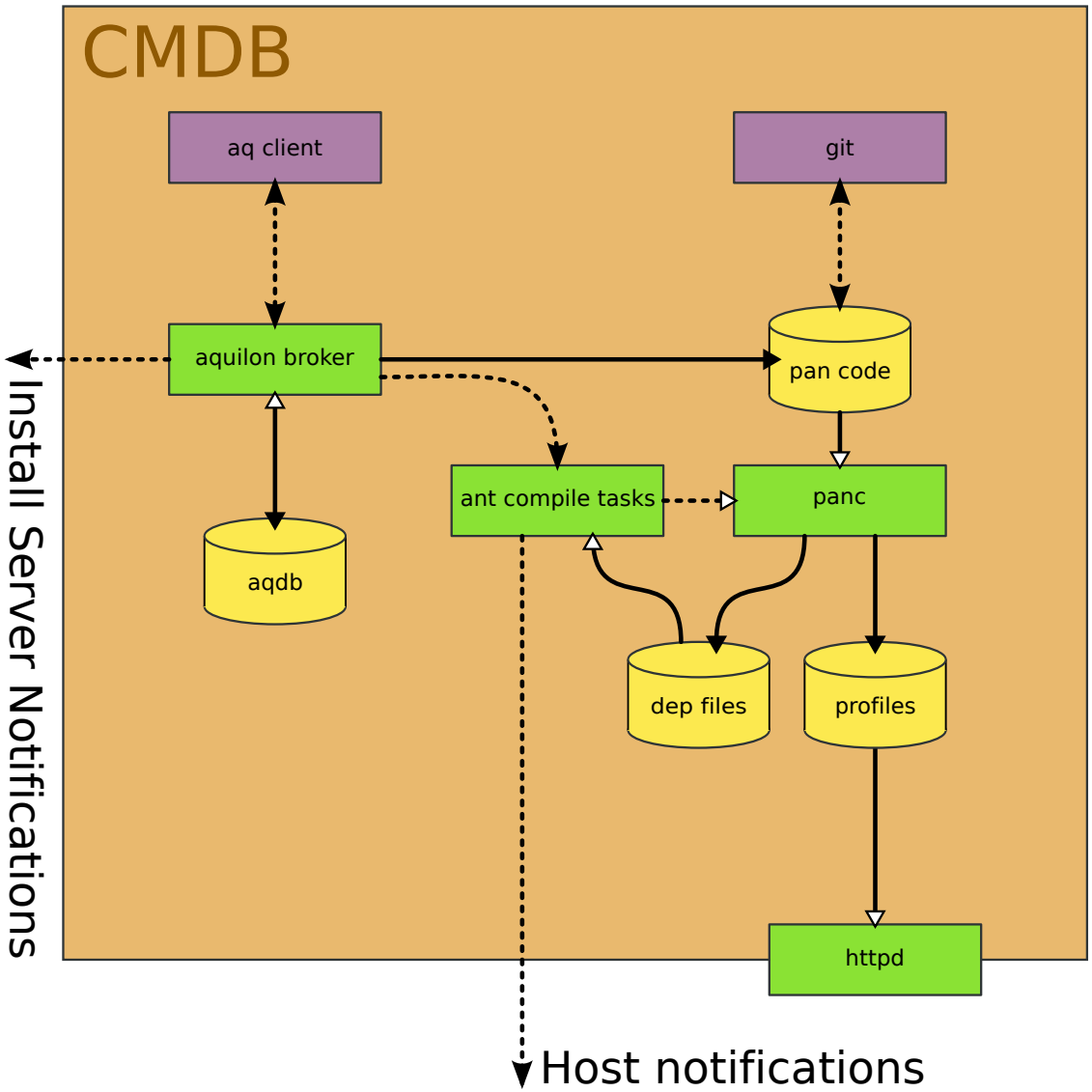
Aquilon

- Third generation configuration management data base
- Builds upon concepts from previous CMDBs
  - But still a paradigm shift
  - Incorporates inventory
  - Provides structure
- Development effort mostly undertaken by Morgan Stanley
  - 85,000+ LOC
  - ~20 contributors

# First impressions

- Git as VCS for Pan code
  - Finally! Proper branching and merging
- Broker daemon running system
  - Owns parts of configuration
  - Role based permissions
- CLI for interaction with broker
  - Make configuration changes
  - Request git branches

# Architecture



# Broker

- Source of **all** power
  - Provides workflow engine
  - Writes Pan code for objects and relationships
  - Owns blessed Git repository
  - Users request branches and work on clones (sandboxes)
  - Allows hosts to be built from sandboxes
- Pure Python
- SQLAlchemy as ORM (very awesome), objects in RDBMS[1]
- REST-ish API for client
  - `/host/www.example.com`
  - `/find/host?personality=webserver`

[1] Many will work, but only PostgreSQL and Oracle are supported.

# Sandboxes

- Production configuration in the `prod` domain
- Branched into sandboxes for development

```
aq add sandbox
    --sandbox new-awesomeness
```

- Creates branch in the broker owned repository
  - Auto-cloned to user's home directory by client

# Sandboxes

- Published for review by others

```
aq publish --sandbox new-awesomeness
```

- Deployed (merged) back into `prod` when ready

```
aq deploy
      --source adamsj/new-awesomeness
      --target prod
```

# Objects

Aquilon provides objects for modelling inventory, high level configuration and the relationships between them.

- Inventory

| | |
|---|---|
| Location | Buildings, Rooms, Racks, Desks... |
| Hardware | Machines, NICs, Drives, CPUs... |
| Network | Switches, Routers, Subnets, Gateways... |

- Configuration

| | |
|---|---|
| Feature | Re-usable block of Pan code configuring something specific |
| Personality | A collection of Features |
| Host | Machine, FQDN, IP, Personality & OS |

Each object has a corresponding `add`, `del`, and `update` command.

# Services and Mappings

- Services
  - Model the concept of a service
  - Particular instances of services
  - Track servers and clients
- Service maps
  - Rules defining which hosts use which instance of which service
  - Rules can be defined based on:
    - Organisation
    - Physical Location
    - Network IP address

# Example

- You have two clusters *arrow* and *angel*:
  - Both have different types of compute node.
  - Each has an NFS server based on the same personality.
  - Each is in a separate subnet.

# Define Services

Define a `nfs` service with an instance for each cluster.

```
aq add service
      --service nfs
     --instance arrow
aq add service
      --service nfs
     --instance angel
```

# Bind Servers

Bind a server to each `nfs` instance.

```
aq bind server
    --service nfs
    --instance arrow
    --hostname snake.example.com
aq bind server
    --service nfs
    --instance angel
    --hostname clockwork.example.com
```

# Add Requirements

Add requirement for `nfs` to both compute node personalities.

```
aq add required service
    --service nfs
  --archetype linux
  --personality gpu-cluster-node
aq add required service
    --service nfs
  --archetype linux
  --personality phi-cluster-node
```

# Map Services

Map service `nfs` based on network subnet.

```
aq map service
     --service cluster-nfs
    --instance arrow
   --networkip 172.16.7.0
aq map service
     --service cluster-nfs
    --instance angel
   --networkip 172.16.12.0
```

# Our Experience

- First site to try and run Aquilon outside Morgan Stanley
  - Lots of work required to generalise
- Running in pre-production now
  - 200 hosts
  - Alongside SCDB
- Using SCDB feels painful by comparison
  - Full migration soon

# Aquilon

- The third generation CMDB for Quattor
- Integrated inventory information
- Provides a framework for configuration code
- Broker is source of ultimate power
- Solution to all your problems

# Thanks

www.quattor.org

www.quattor.org/documentation/2013/10/25/aquilon-site.html

www.github.com/quattor/aquilon