Configuration automation for CERN's new Wi-Fi infrastructure

Quentin Barrand
CERN IT / Communication Services
Agenda

• Configuring Aruba controllers and APs
• Why and how to automate ?
• Interacting with the devices
• Questions & feedback
Our PROD cluster

- **Uplink** (Brocade MLXe routers)
- **Mobility Masters** (master / standby)
- **7240 controllers** (all active)
- **Access Points** (9 models)
How are APs managed?

- Any AP belongs to an AP Group
- The group’s settings are applied to all its APs
  - Transmitted power
  - Allowed channels & channel width
  - Broadcasted SSIDs
  - ...
- But!
  - RF settings *can* also be defined per-AP
  - External antenna gains *must* be defined per-AP
Configuration hierarchy

Mobility Master branch

Managed Devices branch

/

/mm

/mm/mynode

/md

/md/PROD

/md/PROD/controller-1

/md/PROD/controller-2

/md/PROD/controller-3

/md/PROD/controller-4
How to deploy an AP?

- **Simple!**
  - Include the cluster’s IP address as **DHCP option 43**
  - Add an entry in the Mobility Master **specifying the AP Group**
  - Optionally, define AP-specific settings (radio parameters, antenna gains…)

- **Except…**
  - We’re deploying at least 4000 AP
  - We have 3 clusters (lab, pilot, production)
  - How do we make sure that the same rules will be enforced in the future?
CERN Network’s Management System

- One central place of truth: **LANDB**
  - Oracle database with a web front-end
  - Stores the information of all the devices connected to the network

- One main management application: **CFMGR**
  - Uses LANDB data to build the network topology
  - Configures equipments (switches, routers...) and services (DNS, DHCP…) mainly via CLI and SNMP
  - ~200k lines of highly-sophisticated Perl code
Adding data to LANDB

<table>
<thead>
<tr>
<th>External Antenna</th>
<th>RF Profile 2.4 GHz</th>
<th>RF Profile 5 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Gain 2.4 GHz</td>
<td>Name on controller</td>
<td>Name on controller</td>
</tr>
<tr>
<td>Gain 5 GHz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster</th>
<th>AP Group</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name</td>
<td>Name on controller</td>
</tr>
<tr>
<td>Mobility Master IP</td>
<td>RF Profile 2.4 GHz</td>
<td></td>
</tr>
<tr>
<td>Cluster IP</td>
<td>RF Profile 5 GHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>External Antenna</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cluster</td>
<td></td>
</tr>
</tbody>
</table>

17/10/2017
HEPiX Fall/Autumn 2017 Workshop 9
Maintain exceptions only

- Infer the AP group from the building number
  - AP in building 33 $\rightarrow$ 0033_default
  - Implemented in LANDB directly (Oracle view)

- A table to define exceptions
  - 600/*-* $\rightarrow$ 0031_default
  - 31/S-* $\rightarrow$ TEST
  - 40/1-A032 $\rightarrow$ 0040_Small-Office

- Another one to define exceptions at the AP level
  - RF profiles
  - Very specific radio settings
  - External antenna
Exception tables

AP Group Exception
- Building
- Floor
- Room
- AP Group

AP Exception
- Device
- RF Profiles…
- External Antenna
- Radio settings…

Network Device
- Name
- Device type
- Hardware address
- Building
- Floor
- Room
- …

Soft binding
How to deploy an AP? (continued)

• Integrate Aruba into CERN’s NMS
  • Only use the device’s CLI for troubleshooting and engineering
  • Script all repetitive tasks

• Just declare the AP as a network equipment in LANDB
  • Name
  • Manufacturer & model
  • Wired MAC address
  • Location (building and office)
  • IP network to which it is connected

• Optionally, add an AP exception for the AP
• Synchronize the data regularly
NETMGR

- **Python 3** application
  - Very large ecosystem (requests, SQLAlchemy...)
  - Big community at CERN
  - Easier to maintain (vs. Perl)

- ~3.5k SLOC (excluding tests)

- Features
  - Compares and pushes LANDB changes to the Mobility Master
  - Also includes a few cross-cluster show commands
Interacting with the Mobility Master

• It Provides a HTTP REST API!
  • JSON output 😊
  • No need to use expect-like patterns anymore 😊

• Better than plain text, but…
  • The API is not versioned 😞
  • The HTTP return code is always 200 OK 😞
  • The error messages are either missing or useless 😞
  • There is no consistency in the parameter names and returned payloads
  • A lot of interesting endpoints are private / undocumented
CLI vs JSON

(R513-C-CARM-21) #show ap database

AP Database
---------

<table>
<thead>
<tr>
<th>Name</th>
<th>Group</th>
<th>AP Type</th>
<th>IP Address</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>U28-S-AP1-BARI1-13</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.136</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-14</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.149</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-15</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.137</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-19</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.138</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-21</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.139</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-24</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.140</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-27</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.141</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-29</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.142</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-33</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.143</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-36</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.144</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-39</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.145</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-4</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.133</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-40</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.146</td>
<td>...</td>
</tr>
<tr>
<td>U28-S-AP1-BARI1-44</td>
<td>0028_default</td>
<td>325</td>
<td>172.28.18.147</td>
<td>...</td>
</tr>
</tbody>
</table>

$> curl -X GET /
/v1/configuration/showcommand?json=1&command=show+ap+database
HTTP/1.1

```json
{
    "AP Database": [
        {
            "AP Type": "325",
            "Flags": "2",
            "Group": "0028_default",
            "IP Address": "172.28.18.136",
            "Name": "U28-S-AP1-BARI1-13",
            "Standby IP": "188.184.0.152",
            "Status": "Up 5d:10h:12m:38s",
            "Switch IP": "188.184.0.151"
        },
        ...
    ]
}
```
Features

Every 5 minutes
Fetch and store the list of wireless associations (security)

Every 15 minutes
Synchronize AP entries and radio settings

Every 30 minutes
Synchronize antenna gains (requires reading the AP’s boot parameters)

(netmgr) cfmgr/qubarran[1002] netmgr show wifi ap --loc 31/r -u
11 devices match your query.

<table>
<thead>
<tr>
<th>AP name</th>
<th>Model</th>
<th>AP Group</th>
<th>Location</th>
<th>Status</th>
<th>Flags</th>
<th>Zone</th>
<th>2GHz (M/Chan/EIRP/Cli)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U31-S-AP1-BARI1-12</td>
<td>325</td>
<td>0031_default</td>
<td>31/R-030</td>
<td>Up 5d:21h:3m:1s</td>
<td>2</td>
<td>IT-PILOT</td>
<td>APHT / 1 / 7 / 0</td>
</tr>
<tr>
<td>U31-S-AP1-BARI1-14</td>
<td>325</td>
<td>0031_default</td>
<td>31/R-023</td>
<td>Up 5d:21h:31m:32s</td>
<td>2</td>
<td>IT-PILOT</td>
<td>APHT / 6 / 8 / 0</td>
</tr>
<tr>
<td>U31-S-AP1-BARI1-15</td>
<td>325</td>
<td>0031_default</td>
<td>31/R-016</td>
<td>Up 5d:21h:3m:2s</td>
<td>2</td>
<td>IT-PILOT</td>
<td>APHT / 1 / 8 / 1</td>
</tr>
<tr>
<td>U31-S-AP1-BARI1-17</td>
<td>325</td>
<td>0031_default</td>
<td>31/R-208</td>
<td>Up 5d:21h:23m:23s</td>
<td>2</td>
<td>IT-PILOT</td>
<td>APHT / 6 / 8 / 0</td>
</tr>
<tr>
<td>U31-S-AP1-BARI1-19</td>
<td>325</td>
<td>0031_default</td>
<td>31/R-008</td>
<td>Up 5d:21h:40m:26s</td>
<td>2</td>
<td>IT-PILOT</td>
<td>APHT / 11 / 7 / 0</td>
</tr>
</tbody>
</table>

...
Wrap-up

- Deploying thousands of devices definitely requires automation
  - Brand new infrastructure but similar workflow and tools as before
  - Deployment and maintenance teams use NETMGR instead of going to the controllers

- More vendors are implementing APIs
  - Removes a lot of boilerplate code
  - Unfortunately not all are ready for prime time…

- **Python** is a great environment for network-related projects
  - Some components of NETMGR have already been reused
Thank you! お世話になっております
Questions, remarks & thoughts