quattor

# *Administering templates with CDB*

Marco Emilio Poleggi - CERN/INFN-CNAF

Marco.Poleggi@cern.ch

# Outline

quattor

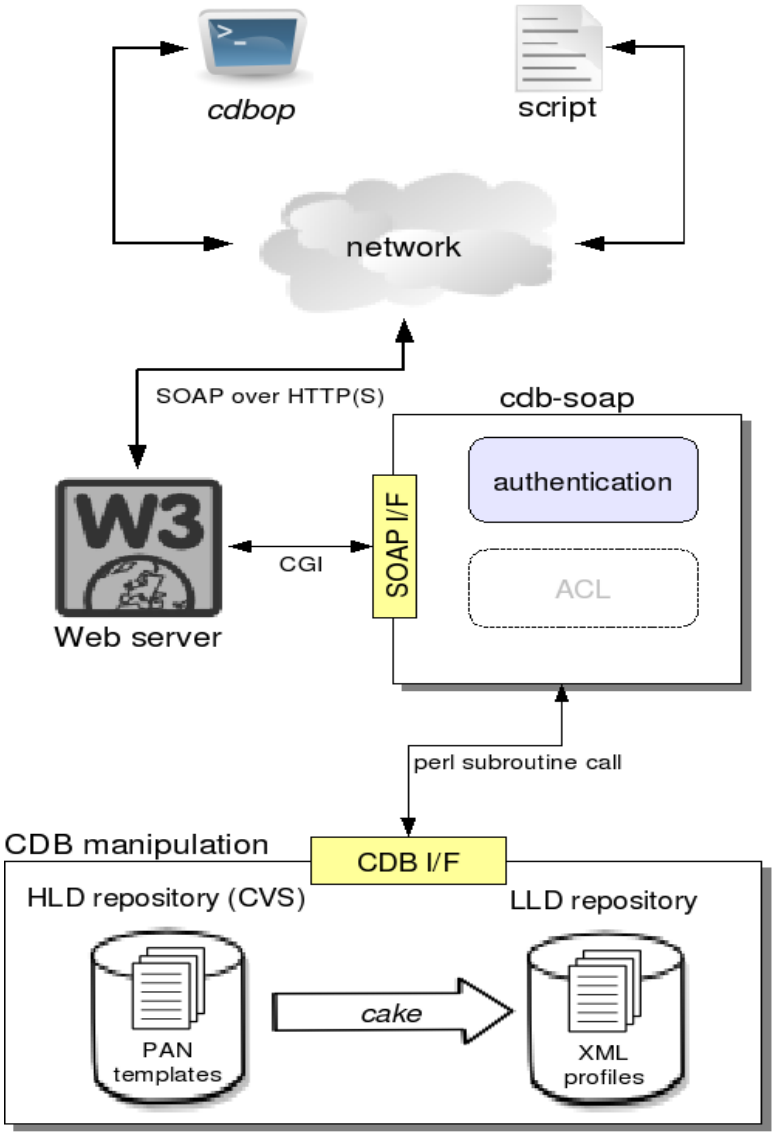
- CDB overview
- Connecting to CDB
- Managing *namespaces*
- Using *groups* and *ACLs*

Material for examples available on agenda page and at:

`http://poleggi.web.cern.ch/poleggi/cdb-ns_example.tgz`

# CDB overview

quattor




## Three-tier architecture

- SOAP client
  - `cdbop` interactive/batch shell
  - Scripts
- SOAP middle-ware
  - Apache + *cdb-soap* CGI
  - stateless: each connection conveys one command
- CDB back-end
  - it's a library, not a server
  - templates compiled via cake
  - templates stored in CVS
  - stateful: partially transactional semantic through “sessions” (not a full-fledged DBMS)
    - ➔ Concurrent sessions

# Outline

quattor

- CDB overview
- Connecting to CDB
  - **cdbop**: The client program
  - Authentication alternatives
- Managing *templates* and *namespaces*
- Using *groups* and *ACLs*

# cdbop: The client program

quattor

- Allows remote management via *ftp*-like commands
  - Log in, check out or add templates, update templates, …, commit, log out
- Shell-like interaction
  - Wildcard expansion
  - Command/name completion
  - Command history
  - Batch-mode processing
  - External command support
- User's actions are encapsulated in *sessions*
  - Sequence of transactions: *do something, commit/rollback, do other, commit…*
  - Session can either see external changes or be isolated
- Disconnected operations
  - Sessions can be left open across several **`cdbop`** runs
  - Time-to-live for garbage collection is configured at the server

# Authentication with encrypted passwords

quattor

- *Default* method. Passwords are
  - stored in a file on the server
  - transferred over *HTTPS*

server

```
# cat /etc/cdb.allow
admin
# cat /etc/cdb.passwd
admin::XyCBC8pPZ2D6o
```

client

```
$ cdbop
quattor CDB CLI: Version 2.0.2
Enter user-name (poleggi): admin
Enter password:
Connecting to https://pc-adc-03...
Welcome to CDB Command Line Interface
Opening session...
Type 'help' for more info
<cdbop: ~/>
```

# Authentication with X.509 certificates

quattor

- Supports *standard* certificates (e.g. grid, but not “proxies”)
  - no password sent over the network
  - authentication made on user's DN

server

```
# cat /etc/cdb.allow
poleggi:/C=CH/O=CERN/OU=GRID/CN=Marco Emilio Poleggi 7603
```

client

```
$ openssl x509 -in ~/.globus/usercert.pem -subject -noout
subject= /C=CH/O=CERN/OU=GRID/CN=Marco Emilio Poleggi
7603
$ cdbop --use-cert
quattor CDB CLI: Version 2.0.2
Enter user-name (poleggi): pol
Enter PEM passphrase:
Connecting to https://pc-adc-03...
Welcome to CDB Command Line Interface
Opening session...
Type 'help' for more info
<cdbop: ~/>
```

My private key is encrypted

# Authentication with Kerberos 5

quattor

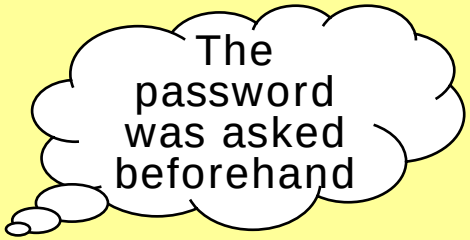
- Full Kerberos environment required
  - no password sent over the network

server

```
# ask_sysadmins... :-P
```

client

```
$ kinit -V
Password for poleggi@CERN.CH:
Authenticated to Kerberos v5
Authenticated to Kerberos v4
$ cdbop --use-krb --server=lxdev23
quattor CDB CLI: Version 2.0.2
Connecting to https://pc-adc-03...
Welcome to CDB Command Line Interface
Opening session...
Type 'help' for more info
<cdbop: ~/>
```



# Outline

quattor

- CDB overview
- Connecting to CDB
- Managing *templates* and *namespaces*
  - The old flat world of templates
  - Everything in its right place
- Using *groups* and *ACLs*

# The old flat world of templates

quattor

- Templates are physically stored and logically shown in the same place
  - class distinction only possible through the name, e.g.: **pro_***, **test_***

client

```
$ cdbop -i
...
<cdbop: ~/> lcd /usr/share/doc/pan-templates/standard
<cdbop: /usr/share/doc/pan-templates/standard> !ls
pro_declaration_functions_filesystem.tpl
pro_declaration_structures.tpl
...
<cdbop: /usr/share/doc/pan-templates/standard> add *
[INFO] 'pro_declaration_functions_filesystem': added
[INFO] 'pro_declaration_functions_general': added
...
<cdbop: /usr/share/doc/pan-templates/standard> list
[WARN] no template matching '*/*' found
pro_declaration_functions_filesystem
pro_declaration_functions_general
...
<cdbop: /usr/share/doc/pan-templates/standard> commit
<cdbop: /usr/share/doc/pan-templates/standard> lcd ~/tmp/cdb-tmp
<cdbop: ~/tmp/cdb-tmp>
```

Isolate mode

External command

Bug ☹

Local check-out

# Adding nodes' profiles

quattor

- *Profile* names are conventionally prefixed with **profile_**

client

```
<cdbop: ~/tmp/cdb-tmp> !vi profile_foo.tpl
<cdbop: ~/tmp/cdb-tmp> add profile_foo.tpl
<cdbop: ~/tmp/cdb-tmp> commit
<cdbop: ~/tmp/cdb-tmp> list
...
pro_declaration_unit
profile_foo
<cdbop: ~/tmp/cdb-tm
<cdbop: ~/tmp/cdb-tm
<cdbop: ~/tmp/cdb-tm
<cdbop: ~/tmp/cdb-tm
[INFO] 'profile_foo'
1.1 2006/06/14 10:17
1.2 2006/06/14 10:21:08
<cdbop: ~/tmp/cdb-tmp> exit
$
```

```
# This is the profile of node "foo"
object template profile_foo;

include pro_declaration_structures;

"/foo" = 'bar';
```

server

```
# cat /var/lib/cdb-lcg-t2-tutorial/lld/xml/profile_foo.xml
<?xml version="1.0" encoding="utf-8"?>
<nlist name="profile" format="pan">
  <string name="foo">bar</string>
</nlist>
```

"LLD" profile

# Namespaces: *everything in its right place*

quattor

- Templates are organized in a directory-like hierarchy. E.g.
  - Two classes of templates with the same structure
    - “testing”
    - “production”
  - A separate namespace for nodes’ profiles

client

```
<cdbop: ~/tmp/cdb-tmp> add_ns production
[INFO] 'production/': namespace added
<cdbop: ~/tmp/cdb-tmp> add_ns testing
[INFO] 'testing/': namespace added
<cdbop: ~/tmp/cdb-tmp> add_ns profiles
[INFO] 'profiles/': namespace added
<cdbop: ~/tmp/cdb-tmp> add_ns production/components
[INFO] 'production/components/': namespace added
<cdbop: ~/tmp/cdb-tmp> add_ns production/hardware
[INFO] 'production/hardware/': namespace added
<cdbop: ~/tmp/cdb-tmp> add_ns testing/components
[INFO] 'testing/components/': namespace added
<cdbop: ~/tmp/cdb-tmp> add_ns testing/hardware
[INFO] 'testing/hardware/': namespace added
```

Need a bootstrap mechanism…

No commit required!

# Namespaces = directory structure in the local check-out

quattor

client

```
<cdbop: ~/tmp/cdb-tmp> list -r
/
pro_declaration_functions_filesystem
...
pro_declaration_units
production/
production/components/
production/hardware/
profile_foo
profiles/
testing/
testing/components/
testing/hardware/
<cdbop: ~/tmp/cdb-tmp> !tree
.
|-- production
|   |-- components
|   `-- hardware
|-- profile_foo.tpl
|-- profiles
`-- testing
    |-- components
    `-- hardware
```

Recursive CDB view

Old stuff still works

local view

# *Adding templates* inside *namespaces*

quattor

Templates have either *full* or *local* path

namespace template

```
[production/]hardware/disk
```

client

```
<cdbop: ~/tmp/cdb-tmp
<cdbop: ~/tmp/cdb-tmp> add testing/hardware/disk.tpl
[INFO] 'testing/hardware/disk': added
<cdbop: ~/tmp/cdb-tmp
<cdbop: ~/tmp/cdb-tmp
[INFO] 'testing/compo
<cdbop: ~/tmp/cdb-tmp> !cp testing/components/bar_config.tpl
         production/c
<cdbop: ~/tmp/cdb-tmp
<cdbop: ~/tmp/cdb-tmp
[INFO] 'production/co
<cdbop: ~/tmp/cdb-tmp> !cp testing/hardware/disk.tpl
         production/h
<cdbop: ~/tmp/cdb-tmp
<cdbop: ~/tmp/cdb-tmp> add production/hardware/disk.tpl
[INFO] 'production/hardware/disk': added
```

```
template hardware/disk;
"/disk"="mydisk testing";
```

```
template components/bar_config;
"/bar_config"="bar config";
"/bar_version"="test";
```

```
template components/bar_config;
"/bar_config"="bar config";
"/bar_version"="production";
```

```
template hardware/disk;
"/disk"="mydisk production";
```

# *Adding nodes' profiles* inside *namespaces*

quattor

- *Panc* resolves template references via the `loadpath` variable
  - No need of prefixing profile names

client

```
<cdbop: ~/tmp/cdb-tmp> 'cp profile_foo_tpl profiles/foo_tpl
<cdbop: ~/tmp/cdb-tmp
<cdbop: ~/tmp/cdb-tmp
[INFO] 'profiles/foo'
<cdbop: ~/tmp/cdb-tmp
```

server

```
# cat /var/lib/cdb-lc
<?xml version="1.0" e
<nlist name="profile"
  <string name="bar_c
  <string name="bar_v
  <list name="current
    <string>testing</
  </list>
  <string name="disk">mydisk testing</string>
  <string name="foo">bar</string>
  <string name="name">foo</string>
</nlist>
```

```
object template foo;

variable loadpath=list("testing");
"/current_loadpath" = loadpath;

include components/bar_con
include hardware/disk;

"/name"="foo";
include pro_declaration_structures;
"/foo" = 'bar';
```

"LLD" profile

# Switching configuration

quattor

- It's a matter of changing the **loadpath** ☺

client

```
<cdbop: ~/tmp/cdb-tmp> !vi profiles/foo.tpl
<cdbop: ~/tmp/cdb-tmp> update profiles/foo.tpl
[INFO] 'profiles/foo'
<cdbop: ~/tmp/cdb-tmp
```

```
object template foo;

variable loadpath=list("production");
"/current_loadpath" = loadpath;

include components/bar_config;
include hardware/disk;

"/name"="foo";
include pro_declaration_structures;
"/foo" = 'bar';
```

server

```
# cat /var/lib/cdb-lc
<?xml version="1.0" e
<nlist name="profile"
  <string name="bar_c
  <string name="bar_v
  <list name="current
    <string>productio
  </list>
  <string name="disk"
  <string name="foo">bar</string>
  <string name="name">foo</string>
</nlist>
```

# Outline

quattor

- CDB overview
- Connecting to CDB
- Managing *templates* and *namespaces*
- Using *groups* and *ACLs*
  - Confining the scope of users' actions

# Groups and ACLs...

quattor

- An *access control list* (ACL) is a statement about what a *user/group* can do with a given *item*
  - *<item>: <user/group>=<what>, <user/group>=<what>, ...*
  - "what" means *permissions*, or *rights*: "read", "write", "admin"
    - semantic similar to Unix file system's permissions
    - "admin" means ability of changing an item's ACLs
  - users may belong to a *group* which is composed of
    - *owners*, who can administer group membership, and
    - normal *members*
  - an "item" can be either a namespace or a full template path
- Two plain *ASCII* files store data for groups and ACLs on the CDB server
  - editable either via `cdbop` or by hand ☺

# ...Groups and ACLs

quattor

- Special user **`admin`**: hard coded, not modifiable
  - So far, we've been connecting as **`admin`**
- Special groups
  - *Default* group **`%all`**: virtual, not modifiable
  - **`%admins`**: each member has the same privileges as **`admin`**, no ACL needed
- Usage scenario
  - Alice and Bob are two *service managers* for the production farm
    - write permission to namespaces **`production/`** and **`profiles/`**
    - read permission to namespaces **`testing/`**
  - Caty and Joe are two *developers* for the production farm
    - write permission to namespaces **`testing/`**

# Setting up groups

quattor

client

```
<cdbop: ~/tmp/cdb-tmp> group_list
%all: admin alice bob caty joe poleggi
<cdbop: ~/tmp/cdb-tmp> group_create %srvman !alice
[INFO] Group '%srvman' created
[INFO] '!alice': added
<cdbop: ~/tmp/cdb-tmp> group_add_member %srvman bob
[INFO] 'bob': added
<cdbop: ~/tmp/cdb-tmp> group_create %dev
[INFO] Group '%dev' created
<cdbop: ~/tmp/cdb-tmp> group_add_member %dev caty joe
[INFO] 'caty': added
[INFO] 'joe': added
<cdbop: ~/tmp/cdb-tmp> group_list
%all: admin alice bob caty joe poleggi
%dev: caty joe
%srvman: !alice bob
```

From /etc/cdb.allow

She's the owner

Normal member

No owner...

server

```
# cat /var/lib/cdb/auth/cdb.groups
%srvman:!alice,bob
%dev:caty,joe
```

# Setting up ACLs

quattor

client

```
<cdbop: ~/tmp/cdb-tmp> acl_set w %srvman production/*/*
         profiles/ profiles/*
production/components/bar_config: %srvman <- w SET
production/hardware/disk: %srvman <- w SET
profiles/: %srvman <- w SET
profiles/foo: %srvman <- w SET
<cdbop: ~/tmp/cdb-tmp> acl_set r %srvman testing/*/*
testing/components/bar_config: %srvman <- r SET
testing/hardware/disk: %srvman <- r SET
<cdbop: ~/tmp/cdb-tmp> acl_set w %dev testing/*/*
testing/components/bar_config: %dev <- w SET
testing/hardware/disk: %dev <- w SET
<cdbop: ~/tmp/cdb-tmp> acl_get
production/components/bar_config: %srvman -> rw
production/hardware/disk: %srvman -> rw
profiles/: %srvman -> rw
profiles/foo: %srvman -> rw
testing/components/bar_config: %dev -> rw, %srvman -> r
testing/hardware/disk: %dev -> rw, %srvman -> r
```

No access to intermediate namespaces

server

```
# cat /var/lib/cdb/auth/cdb.acls
production/components/bar_config:%srvman=rw
...
testing/hardware/disk:%srvman=r,%dev=rw
```

No wildcards here!

# Logging in as non-privileged users

quattor

client

```
$ cdbop -i
quattor CDB CLI: Version 2.0.3
Enter user-name (poleggi): alice
...
<cdbop: ~/tmp/cdb-tmp2> get profiles/foo
[INFO] 'profiles/foo.tpl': received
<cdbop: ~/tmp/cdb-tmp2> !vi profiles/foo.tpl
<cdbop: ~/tmp/cdb-tmp2> update profiles/foo.tpl
[INFO] 'profiles/foo': updated
<cdbop: ~/tmp/cdb-tmp2> get testing/hardware/disk
[INFO] 'testing/hardware/disk.tpl': received
<cdbop: ~/tmp/cdb-tmp2> update testing/hardware/disk.tpl
[ERROR] error(s) found:
[ERROR] 'testing/hardware/disk': permission denied
```

Alice logs in

A local directory is created

Ouch!

client

```
$ cdbop -i
quattor CDB CLI: Version 2.0.3
Enter user-name (poleggi): joe
...
<cdbop: ~/tmp/cdb-tmp2> get production/components/bar_config
[ERROR] error(s) found:
[ERROR] 'production/components/bar_config': permission denied
<cdbop: ~/tmp/cdb-tmp2> acl_check r joe
         production/components/bar_config
production/components/bar_config: r ?? joe: DENIED
```

Joe logs in

Can't even read it

...indeed

# References

quattor

- User commands
  - On the client: `$ man cdbop`
  - On the server: `/usr/share/doc/cdb-soap-server-<version>/ACLs-documentation.html`
  - Appendix H of *Quattor Installation and User Guide*
- Writing templates
  - PAN Language Specifications
- Quattor tutorials at `http://quattor.web.cern.ch/quattor/documentation/tutorials/`


**http://quattor.org**