
Google ganeti

Ganeti

Ganeti Core Team - Google
LISA '13 - 5 Nov 2013

Latest version of these slides

Please find the latest version of these slides at:

https://code.google.com/p/ganeti/wiki/LISA2013

Configuration Daemon (ConfD)

- Guido Trotter <ultrotter@google.com>
- Helga Velroyen <helgav@google.com>
- (Slides by Michele Tartara <mtartara@google.com>)

Once upon a `t` ...

For `t < 2.1`

- Configuration only available on master candidates
- Few selected values replicated with Ssconf
  - Small pieces of config in text files on all the nodes
  - Doesn't scale
- Need a way to access config from other nodes
  - Scalable
  - No single point of failure (so, no RAPI)

Enters ConfD

- Provides information from `config.data`
- Read-only
- Distributed
  - Multiple daemons running on master candidates
  - Accessible from all the nodes through ConfD protocol
  - Resilient to failures
- Optional

What info does it provide?

Replies to simple queries:

- Ping
- Master IP
- Node role
- Node primary IP
- Master candidates primary IPs
- Instance IPs
- Node primary IP from Instance primary IP
- Node DRBD minors
- Node instances

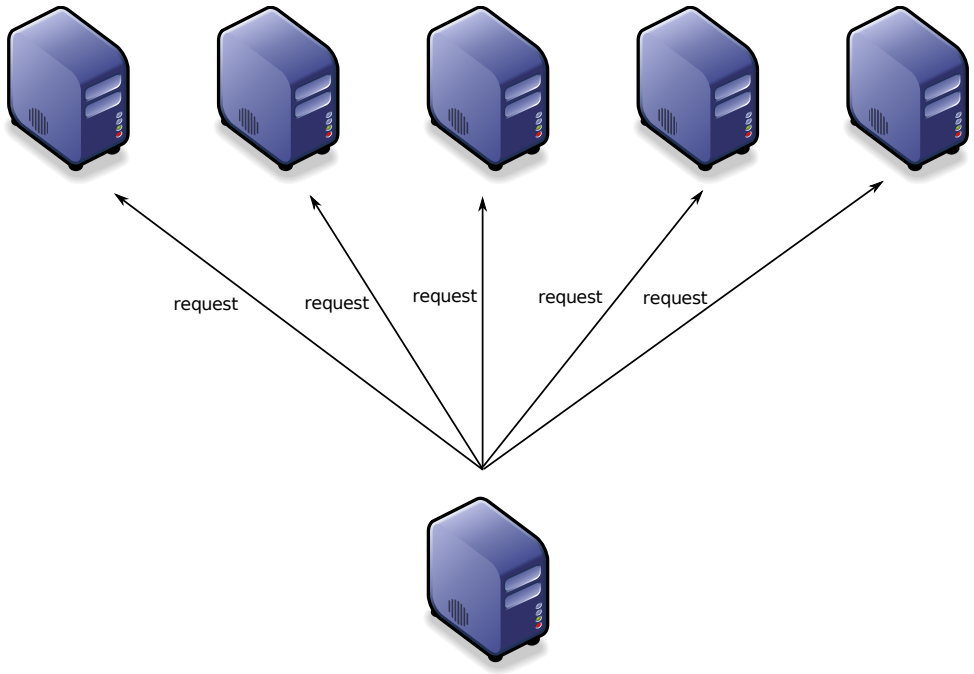
ConfD protocol

General description

- UDP (port 1814)
- keyed-Hash Message Authentication Code (HMAC) authentication
  - Pre-shared, cluster wide key
  - Generated at cluster-init
  - Root-only readable
- Timestamp
  - Checked (± 2.5 mins) to prevent replay attacks
  - Used as HMAC salt
- Queries made to any subset of master candidates
- Timeout
- Maximum number of expected replies

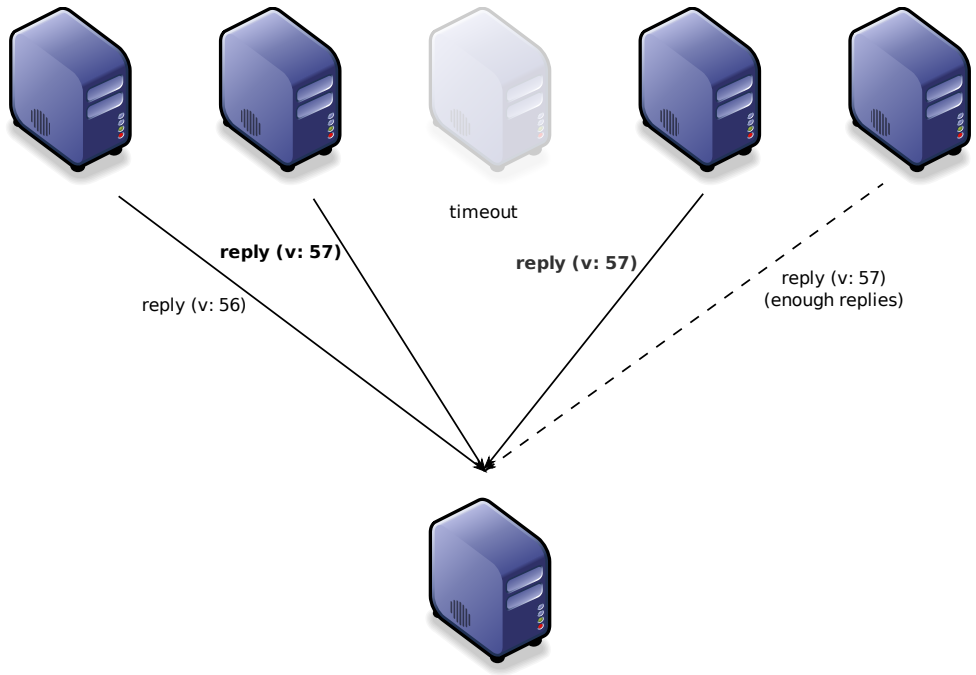
Confd protocol

Request/Reply



Confd protocol

Request/Reply



ConfD protocol

Request

CONFD

```
plj0{
  "msg": "{\"type\": 1,
           \"rsalt\": \"9aa6ce92-8336-11de-af38-001d093e835f\",
           \"protocol\": 1,
           \"query\": \"node1.example.com\"}\n",
  "salt": "1249637704",
  "hmac": "4a4139b2c3c5921f7e439469a0a45ad200aead0f"
}
```

- `plj0:` fourcc detailing the message content (PLain Json 0)
- `hmac:` HMAC signature of salt+msg with the cluster hmac key

ConfD protocol

Request

CONFD

```
plj0{
  "msg": "{\"type\": 1,
           \"rsalt\": \"9aa6ce92-8336-11de-af38-001d093e835f\",
           \"protocol\": 1,
           \"query\": \"node1.example.com\"}\n",
  "salt": "1249637704",
  "hmac": "4a4139b2c3c5921f7e439469a0a45ad200aead0f"
}
```

- `msg:` JSON-encoded query
  - `protocol:` ConfD protocol version (=1)
  - `type:` What to ask for (`CONFD_REQ_*` constants)
  - `query:` additional parameters
  - `rsalt:` response salt == UUID identifying the request

ConfD protocol

Reply

CONFD

```
plj0{
  "msg": "{\"status\": 0,
           \"answer\": 0,
           \"serial\": 42,
           \"protocol\": 1}\n",
  "salt": "9aa6ce92-8336-11de-af38-001d093e835f",
  "hmac": "aaeccc0dff9328fdf7967cb600b6a80a6a9332af"
}
```

- `salt:` the rsalt of the query
- `hmac:` hmac signature of salt+msg

ConfD protocol

Reply

CONFD

```
plj0{
  "msg": "{\"status\": 0,
           \"answer\": 0,
           \"serial\": 42,
           \"protocol\": 1}\n",
  "salt": "9aa6ce92-8336-11de-af38-001d093e835f",
  "hmac": "aaeccc0dff9328fdf7967cb600b6a80a6a9332af"
}
```

- `msg:` JSON-encoded answer
  - `protocol:` protocol version (=1)
  - `status:` 0=ok; 1=error
  - `answer:` query-specific reply
  - `serial:` version of `config.data`

Ready-made clients

The protocol is simple, but clients are simpler

- Ready to use ConfD clients
  - Python
    - `lib/confd/client.py`
  - Haskell
    - Since Ganeti 2.7
    - `src/Ganeti/ConfD/Client.hs`
    - `src/Ganeti/ConfD/ClientFunctions.hs`

Expanding ConfD capabilities

- Currently not so many queries are supported
- Easy to add new ones
  - Just add a new query type in the constants list
  - ...and extend the `buildResponse` function (`src/Ganeti/Confd/Server.hs` to reply to it in the appropriate way

Conclusion

- More info in `doc/design-2.1.rst`
- Future work
  - More queries can be easily added as needed
  - Management of the configuration (on master) moved to a separate daemon from masterd

Thank You!

Questions?