CTDB, you have changed!

Martin Schwenke <martin@meltin.net>

Samba Team
IBM (Australia Development Laboratory, Linux Technology Center)

SambaXP 2018

Overview

- Configuration
- Event handling
- Service management
- IP failover

The journey

The journey

- `conntrackd` [M]: March-October 2017, self-contained

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction
- `failoverd` [M]: December 2017, false start

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction
- `failoverd` [M]: December 2017, false start
- `conf` [A↑]: January-April 2018, abstraction around `tini`

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction
- `failoverd` [M]: December 2017, false start
- `conf` [A↑]: January-April 2018, abstraction around `tini`
- `cmdline` [A↑]: April 2018, command and option abstraction

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction
- `failoverd` [M]: December 2017, false start
- `conf` [A↑]: January-April 2018, abstraction around `tini`
- `cmdline` [A↑]: April 2018, command and option abstraction
- `path` [A↑]: May 2018, file/directory path abstraction

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction
- `failoverd` [M]: December 2017, false start
- `conf` [A↑]: January-April 2018, abstraction around `tini`
- `cmdline` [A↑]: April 2018, command and option abstraction
- `path` [A↑]: May 2018, file/directory path abstraction
- `ctdbd+conf` [M↑]: February-May 2018, tests, scripts, `ctdbd`

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction
- `failoverd` [M]: December 2017, false start
- `conf` [A↑]: January-April 2018, abstraction around `tini`
- `cmdline` [A↑]: April 2018, command and option abstraction
- `path` [A↑]: May 2018, file/directory path abstraction
- `ctdbd+conf` [M↑]: February-May 2018, tests, scripts, `ctdbd`
- `eventd` [A]: February-May 2018, multi-component

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction
- `failoverd` [M]: December 2017, false start
- `conf` [A↑]: January-April 2018, abstraction around `tini`
- `cmdline` [A↑]: April 2018, command and option abstraction
- `path` [A↑]: May 2018, file/directory path abstraction
- `ctdbd+conf` [M↑]: February-May 2018, tests, scripts, `ctdbd`
- `eventd` [A]: February-May 2018, multi-component
- `serviced` [A]: February-May 2018, simple, just services

The journey

- `conntrackd` [M]: March-October 2017, self-contained
- `sock_client` [A↑]: June 2017, tool abstraction
- `failoverd` [M]: December 2017, false start
- `conf` [A↑]: January-April 2018, abstraction around `tini`
- `cmdline` [A↑]: April 2018, command and option abstraction
- `path` [A↑]: May 2018, file/directory path abstraction
- `ctdbd+conf` [M↑]: February-May 2018, tests, scripts, `ctdbd`
- `eventd` [A]: February-May 2018, multi-component
- `serviced` [A]: February-May 2018, simple, just services
- `failoverd` [M+A]: May- 2018

Configuration

Configuration — How it was

- `ctdbd.conf`: shell variables

Configuration — How it was

- `ctdbd.conf`: shell variables
- Daemon command-line options

Configuration — How it was

- `ctdbd.conf`: shell variables
- Daemon command-line options
- Script options: shell variables in `ctdbd.conf`

Configuration — How it was

- `ctdbd.conf`: shell variables
- Daemon command-line options
- Script options: shell variables in `ctdbd.conf`
- Tunable options: shell variables prefixed with `CTDB_SET_`

Configuration — How it was

- `ctdbd.conf`: shell variables
- Daemon command-line options
- Script options: shell variables in `ctdbd.conf`
- Tunable options: shell variables prefixed with `CTDB_SET_`

- `ctdbd_wrapper`: `ctdbd.conf` → `ctdbd` command-line

Configuration — How it was

- `ctdbd.conf`: shell variables
- Daemon command-line options
- Script options: shell variables in `ctdbd.conf`
- Tunable options: shell variables prefixed with `CTDB_SET_`

- `ctdbd_wrapper`: `ctdbd.conf` → `ctdbd` command-line
- `00.ctdb`: `ctdbd.conf` → `ctdb setvar ...`

Configuration — How it was

- `ctdbd.conf`: shell variables
- Daemon command-line options
- Script options: shell variables in `ctdbd.conf`
- Tunable options: shell variables prefixed with `CTDB_SET_`

- `ctdbd_wrapper`: `ctdbd.conf` → `ctdbd` command-line
- `00.ctdb`: `ctdbd.conf` → `ctdb setvar ...`

- Tests assumed shell variable-based configuration

Configuration — How it was

- `ctdbd.conf`: shell variables
- Daemon command-line options
- Script options: shell variables in `ctdbd.conf`
- Tunable options: shell variables prefixed with `CTDB_SET_`

- `ctdbd_wrapper`: `ctdbd.conf` → `ctdbd` command-line
- `00.ctdb`: `ctdbd.conf` → `ctdb setvar ...`

- Tests assumed shell variable-based configuration
- Nasty hacks!

Configuration — How it was — Example

ctdbd.conf:

```
CTDB_RECOVERY_LOCK="/shared/.ctdb/recovery_lock"

CTDB_LOGGING="syslog"

CTDB_PUBLIC_ADDRESSES="/etc/ctdb/public_addresses"

CTDB_MANAGES_SAMBA="yes"

CTDB_SET_TDBMutexEnabled=1
```

Configuration — How it is

- `ctdb.conf`: Samba-style, daemon configuration

Configuration — How it is

- `ctdb.conf`: Samba-style, daemon configuration
- `script.options`: shell variables

Configuration — How it is

- `ctdb.conf`: Samba-style, daemon configuration
- `script.options`: shell variables
- `ctdb.tunables`: still `00.ctdb`, legacy

Configuration — How it is

- `ctdb.conf`: Samba-style, daemon configuration
- `script.options`: shell variables
- `ctdb.tunables`: still `00.ctdb`, legacy
- `ctdb-config`: allows scripts to get daemon options

Configuration — How it is

- `ctdb.conf`: Samba-style, daemon configuration
- `script.options`: shell variables
- `ctdb.tunables`: still `00.ctdb`, legacy
- `ctdb-config`: allows scripts to get daemon options
- `ctdb-path`: config, socket, PID file locations

Configuration — How it is

- `ctdb.conf`: Samba-style, daemon configuration
- `script.options`: shell variables
- `ctdb.tunables`: still `00.ctdb`, legacy
- `ctdb-config`: allows scripts to get daemon options
- `ctdb-path`: config, socket, PID file locations

- Test hooks via `CTDB_TEST_MODE`

Configuration — How it is

- `ctdb.conf`: Samba-style, daemon configuration
- `script.options`: shell variables
- `ctdb.tunables`: still `00.ctdb`, legacy
- `ctdb-config`: allows scripts to get daemon options
- `ctdb-path`: config, socket, PID file locations

- Test hooks via `CTDB_TEST_MODE`
- Also `CTDB_SOCKET`, `CTDB_PIDFILE`

Configuration — How it is — Example

ctdb.conf:

```
    [logging]
            location = syslog

    [cluster]
            recovery lock = /shared/.ctdb/recovery_lock
```

50.samba.options (or script.options):

```
    CTDB_MANAGES_SAMBA="yes"
```

ctdb.tunables:

```
    TDBMutexEnabled=1
```

Configuration — How it will be

- `ctdb.conf`: Samba-style, daemon configuration
- `script.options`: **rarely used, no** `CTDB_MANAGES_...`
- **Tunable options move to** `ctdb.conf`, `script.options`
- `ctdb-config`: allows scripts to get daemon options
- `ctdb-path`: config, socket, PID file locations

- Test hooks via `CTDB_TEST_MODE`

Configuration — How it will be — Example

ctdb.conf:

```
[logging]
        location = syslog

[cluster]
        recovery lock = /shared/.ctdb/recovery_lock

[database]
        tdb mutexes = true
```

Configuration — What is gone?

CTDB_PIDFILE	--pidfile
CTDB_SOCKET	--socket
CTDB_NODES	--nlist
CTDB_PUBLIC_ADDRESSES	--public-addresses
CTDB_EVENT_SCRIPT_DIR	--event-script-dir
CTDB_NOTIFY_SCRIPT	--notification-script
CTDB_PUBLIC_INTERFACE	--public-interface
CTDB_MAX_PERSISTENT_CHECK_ERRORS	--max-persistent-check-errors
CTDB_MANAGED_SERVICES	...
CTDB_SHUTDOWN_TIMEOUT	All ctdbd command-line options
CTDB_NODES_FILE (onnode)	except -i | --interactive

CTDB_DBDIR=tmpfs

CTDB_SUPPRESS_COREFILE
CTDB_MAX_OPEN_FILES

Configuration — What will be gone?

- `CTDB_MANAGES_...`
- `ctdb.tunables`
- `ctdbd_wrapper`
- `CTDB_SOCKET`, `CTDB_PIDFILE` (from testing)

Event management

Event management — How it was

- Directory: `events.d/`

Event management — How it was

- Directory: `events.d/`
- Script: `50.samba`

Event management — How it was

- Directory: `events.d/`
- Script: `50.samba`
- Service: `CTDB_MANAGES_SAMBA=yes`

Event management — How it was

- Directory: `events.d/`
- Script: `50.samba`
- Service: `CTDB_MANAGES_SAMBA=yes`
- Configuration: `ctdbd.conf`

Event management — How it was

- Directory: `events.d/`
- Script: `50.samba`
- Service: `CTDB_MANAGES_SAMBA=yes`
- Configuration: `ctdbd.conf`
- Notifications: `CTDB_NOTIFY_SCRIPT`, `notify.sh`

Event management — How it is

- Directory: `events/legacy/`

Event management — How it is

- Directory: `events/legacy/`
- Script: `50.samba.script`

Event management — How it is

- Directory: `events/legacy/`
- Script: `50.samba.script`
- Service: `CTDB_MANAGES_SAMBA=yes`

Event management — How it is

- Directory: `events/legacy/`
- Script: `50.samba.script`
- Service: `CTDB_MANAGES_SAMBA=yes`
- Configuration: `50.samba.options` (or `script.options`)

Event management — How it is

- Directory: `events/legacy/`
- Script: `50.samba.script`
- Service: `CTDB_MANAGES_SAMBA=yes`
- Configuration: `50.samba.options` (or `script.options`)
- Notifications: `events/notification/`

Event management — How it will be

- Directory: `events/{cluster,service,failover}/`
- Script: `50.samba.script`
- Service: **Enable the script!**
- Configuration: `50.samba.options` (or `script.options`)
- Notifications: `events/notification/`

Event management — How it was — Bonus example

events.d/11.natgw:

```
#!/bin/sh
...
```

ctdbd.conf:

```
CTDB_NATGW_PUBLIC_IP=10.1.1.121/24
CTDB_NATGW_PUBLIC_IFACE=eth1
CTDB_NATGW_DEFAULT_GATEWAY=10.1.1.254
CTDB_NATGW_PRIVATE_NETWORK=192.168.1.0/24
CTDB_NATGW_NODES=/etc/ctdb/natgw_nodes
```

natgw_nodes:

```
192.168.1.1
192.168.1.2
192.168.1.3
```

Event management — How it will be – Bonus example

```
events/failover/40.natgw.script:

    #!/bin/sh
    ...

events/failover/40.natgw.options:

    CTDB_NATGW_PUBLIC_IP=10.1.1.121/24
    CTDB_NATGW_PUBLIC_IFACE=eth1
    CTDB_NATGW_DEFAULT_GATEWAY=10.1.1.254
    CTDB_NATGW_PRIVATE_NETWORK=192.168.1.0/24

events/failover/40.natgw.nodes:

    192.168.1.1
    192.168.1.2
    192.168.1.3
```

Event management — How it could be – Bonus example

events/failover/40.natgw.script:

```
#!/bin/sh
...
```

events/failover/40.natgw.options:

```
CTDB_NATGW_PUBLIC_IP=10.1.1.121/24
CTDB_NATGW_PUBLIC_IFACE=eth1
CTDB_NATGW_DEFAULT_GATEWAY=10.1.1.254
CTDB_NATGW_PRIVATE_NETWORK=192.168.1.0/24
```

Event management — eventd

- New `eventd`
- Handles multiple components
- . . . and notifications

Service management

Service management

- Currently have `ctdb_eventd` and event scripts
- Subtract IP failover handling to leave services
- Replaceable with 3rd party subsystem (e.g. Pacemaker)?

Service management — events

startup	start services
shutdown	stop services
monitor	check service health
reconfigure	post-failover, current `ipreallocated`
failover	kick `failoverd` (maybe)

Service management — integration

Service management — integration

- Shut down services on `INACTIVE`

Service management — integration

- Shut down services on `INACTIVE`
- Kick `failoverd` on `UNHEALTHY` $\leftrightarrow$ `HEALTHY`

IP failover

IP failover

- `failoverd`

IP failover

- `failoverd`
- Handles assignment & hosting of:
  - addresses e.g. public IP addresses
  - roles e.g. NAT gateway master

IP failover

- `failoverd`
- Handles assignment & hosting of:
  - addresses e.g. public IP addresses
  - roles e.g. NAT gateway master
- Fully event script driven

IP failover

- `failoverd`
- Handles assignment & hosting of:
  - addresses e.g. public IP addresses
  - roles e.g. NAT gateway master
- Fully event script driven
- State can be stored in replicated database

IP failover

- `failoverd`
- Handles assignment & hosting of:
  - addresses e.g. public IP addresses
  - roles e.g. NAT gateway master
- Fully event script driven
- State can be stored in replicated database
- Repeated failure causes node to be banned

IP failover — event scripts

10.pubip	public IP address handling, policy routing
20.lvs	Linux Virtual Server support
30.static_routes	existing simple static route management
40.natgw	existing NAT gateway support

IP failover — monitoring

- Monitor loop

IP failover — monitoring

- Monitor loop
- Network interface(s) down does not cause `UNHEALTHY`

IP failover — monitoring

- Monitor loop
- Network interface(s) down does not cause `UNHEALTHY`
- Any change in monitor state causes failover

IP failover — monitoring

- Monitor loop
- Network interface(s) down does not cause `UNHEALTHY`
- Any change in monitor state causes failover
- `CTDB_PARTIALLY_ONLINE_INTERFACES` no longer optional!

IP failover — monitoring

- Monitor loop
- Network interface(s) down does not cause `UNHEALTHY`
- Any change in monitor state causes failover
- `CTDB_PARTIALLY_ONLINE_INTERFACES` no longer optional!
- Monitoring is unusual: continues after 1st failure

IP failover — monitoring

- Monitor loop
- Network interface(s) down does not cause `UNHEALTHY`
- Any change in monitor state causes failover
- `CTDB_PARTIALLY_ONLINE_INTERFACES` no longer optional!
- Monitoring is unusual: continues after 1st failure
- Monitoring updates available addresses/roles...

IP failover — monitoring

- Monitor loop
- Network interface(s) down does not cause `UNHEALTHY`
- Any change in monitor state causes failover
- `CTDB_PARTIALLY_ONLINE_INTERFACES` no longer optional!
- Monitoring is unusual: continues after 1st failure
- Monitoring updates available addresses/roles...
- ...in that replicated database

IP failover — events

Master node runs **calculate** event. . .

. . . then, on all active nodes, in lock-step:

IP failover — events

Master node runs **calculate** event. . .

. . . then, on all active nodes, in lock-step:

prepare desired state, cache locally

IP failover — events

Master node runs **calculate** event. . .

. . . then, on all active nodes, in lock-step:

prepare desired state, cache locally

release addresses/roles

IP failover — events

Master node runs **calculate** event. . .

. . . then, on all active nodes, in lock-step:

- prepare desired state, cache locally
- release addresses/roles
- take addresses/roles

IP failover — events

Master node runs **calculate** event. . .

. . . then, on all active nodes, in lock-step:

- prepare desired state, cache locally
- release addresses/roles
- take addresses/roles
- finalise tweaks, routing changes, . . .

CTDB daemon

CTDB daemon — what remains?

CTDB daemon — what remains?

- Node transport/coordination
- Cluster membership
- Cluster leadership?
- Databases
- Eventually separate out database daemon(s)
- `ctdbd` handles startup/shutdown
- . . . and node inactive/active transitions

Legal Statement

- This work represents the view of the authors and does not necessarily represent the view of IBM.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.

Questions?