Running And Troubleshooting A Samba/CTDB Cluster

A Tutorial At sambaXP 2011

Michael Adam

obnox@samba.org

SerNet / Samba Team

2011-05-09

Welcome to enjoy today's

Starting Points

- given: file system cluster
- want: offer cluster file system on the network (Samba)
- want: all-active
- Samba daemons on nodes need to act as *one* Samba server
- $\Rightarrow$ problems with Samba IPC between nodes
- $\Rightarrow$ need to share some persistent data
- middleware CTDB makes Samba clusterable

Starting Points

- given: file system cluster
- want: offer cluster file system on the network (Samba)
- want: all-active
- Samba daemons on nodes need to act as *one* Samba server
- $\Rightarrow$ problems with Samba IPC between nodes
- $\Rightarrow$ need to share some persistent data
- middleware CTDB makes Samba clusterable

Starting Points

- given: file system cluster
- want: offer cluster file system on the network (Samba)
- want: all-active
- Samba daemons on nodes need to act as *one* Samba server
- $\Rightarrow$ problems with Samba IPC between nodes
- $\Rightarrow$ need to share some persistent data
- middleware CTDB makes Samba clusterable

Starting Points

- given: file system cluster
- want: offer cluster file system on the network (Samba)
- want: all-active
- Samba daemons on nodes need to act as *one* Samba server
- $\Rightarrow$ problems with Samba IPC between nodes
- $\Rightarrow$ need to share some persistent data
- middleware CTDB makes Samba clusterable

Starting Points

- given: file system cluster
- want: offer cluster file system on the network (Samba)
- want: all-active
- Samba daemons on nodes need to act as *one* Samba server
- $\Rightarrow$ problems with Samba IPC between nodes
- $\Rightarrow$ need to share some persistent data
- middleware CTDB makes Samba clusterable

Starting Points

- given: file system cluster
- want: offer cluster file system on the network (Samba)
- want: all-active
- Samba daemons on nodes need to act as *one* Samba server
- $\Rightarrow$ problems with Samba IPC between nodes
- $\Rightarrow$ need to share some persistent data
- middleware CTDB makes Samba clusterable

Starting Points

- given: file system cluster
- want: offer cluster file system on the network (Samba)
- want: all-active
- Samba daemons on nodes need to act as *one* Samba server
- $\Rightarrow$ problems with Samba IPC between nodes
- $\Rightarrow$ need to share some persistent data
- middleware CTDB makes Samba clusterable

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mappings: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

Challenges For Samba

- IPC: messaging (`messages.tdb` and signals)
- IPC: share volatile session data:
  - SMB sessions: `sessionid.tdb`
  - share connections: `connections.tdb`
  - share modes (open files): `locking.tdb`
  - byte range locks: `brlock.tdb`
- share persistent data:
  - user database: `passdb.tdb`
  - domain join information: `secrets.tdb`
  - id mapping tables: `winbindd_idmap.tdb`
  - registry: `registry.tdb`
  - group mapping: `group_mapping.tdb`

CTDB provides

- clustered TDB implementation for volatile and persistent data
- volatile implementation scales better than general purpose cluster DBs
- messaging for Samba between nodes
- cluster service management (start/stop/monitor)

CTDB provides

- clustered TDB implementation for volatile and persistent data
- volatile implementation scales better than general purpose cluster DBs
- messaging for Samba between nodes
- cluster service management (start/stop/monitor)

CTDB provides

- clustered TDB implementation for volatile and persistent data
- volatile implementation scales better than general purpose cluster DBs
- messaging for Samba between nodes
- cluster service management (start/stop/monitor)

CTDB provides

- clustered TDB implementation for volatile and persistent data
- volatile implementation scales better than general purpose cluster DBs
- messaging for Samba between nodes
- cluster service management (start/stop/monitor)

CTDB Design – General

- one daemon `ctdbd` on each node (and temporary forks)
- `smbd` talks to local ctdbd for messaging and TDB access
- `ctdbd` handles metadata of TDBs via the network
- `ctdbd` keeps local TDB copy (LTDB) for fast data reads/writes
- the actual record read and write ops are directly to the LTDB
- normal and persistent TDBs are handled differently
- Note: each single samba client connection is served by a single node and *not* spread across multiple nodes.

CTDB Design – General

- one daemon `ctdbd` on each node (and temporary forks)
- `smbd` talks to local ctdbd for messaging and TDB access
- `ctdbd` handles metadata of TDBs via the network
- `ctdbd` keeps local TDB copy (LTDB) for fast data reads/writes
- the actual record read and write ops are directly to the LTDB
- normal and persistent TDBs are handled differently
- Note: each single samba client connection is served by a single node and *not* spread across multiple nodes.

CTDB Design – General

- one daemon `ctdbd` on each node (and temporary forks)
- `smbd` talks to local ctdbd for messaging and TDB access
- `ctdbd` handles metadata of TDBs via the network
- `ctdbd` keeps local TDB copy (LTDB) for fast data reads/writes
- the actual record read and write ops are directly to the LTDB
- normal and persistent TDBs are handled differently
- Note: each single samba client connection is served by a single node and *not* spread across multiple nodes.

CTDB Design – General

- one daemon `ctdbd` on each node (and temporary forks)
- `smbd` talks to local ctdbd for messaging and TDB access
- `ctdbd` handles metadata of TDBs via the network
- `ctdbd` keeps local TDB copy (LTDB) for fast data reads/writes
- the actual record read and write ops are directly to the LTDB
- normal and persistent TDBs are handled differently
- Note: each single samba client connection is served by a single node and *not* spread across multiple nodes.

CTDB Design – General

- one daemon `ctdbd` on each node (and temporary forks)
- `smbd` talks to local ctdbd for messaging and TDB access
- `ctdbd` handles metadata of TDBs via the network
- `ctdbd` keeps local TDB copy (LTDB) for fast data reads/writes
- the actual record read and write ops are directly to the LTDB
- normal and persistent TDBs are handled differently
- Note: each single samba client connection is served by a single node and *not* spread across multiple nodes.

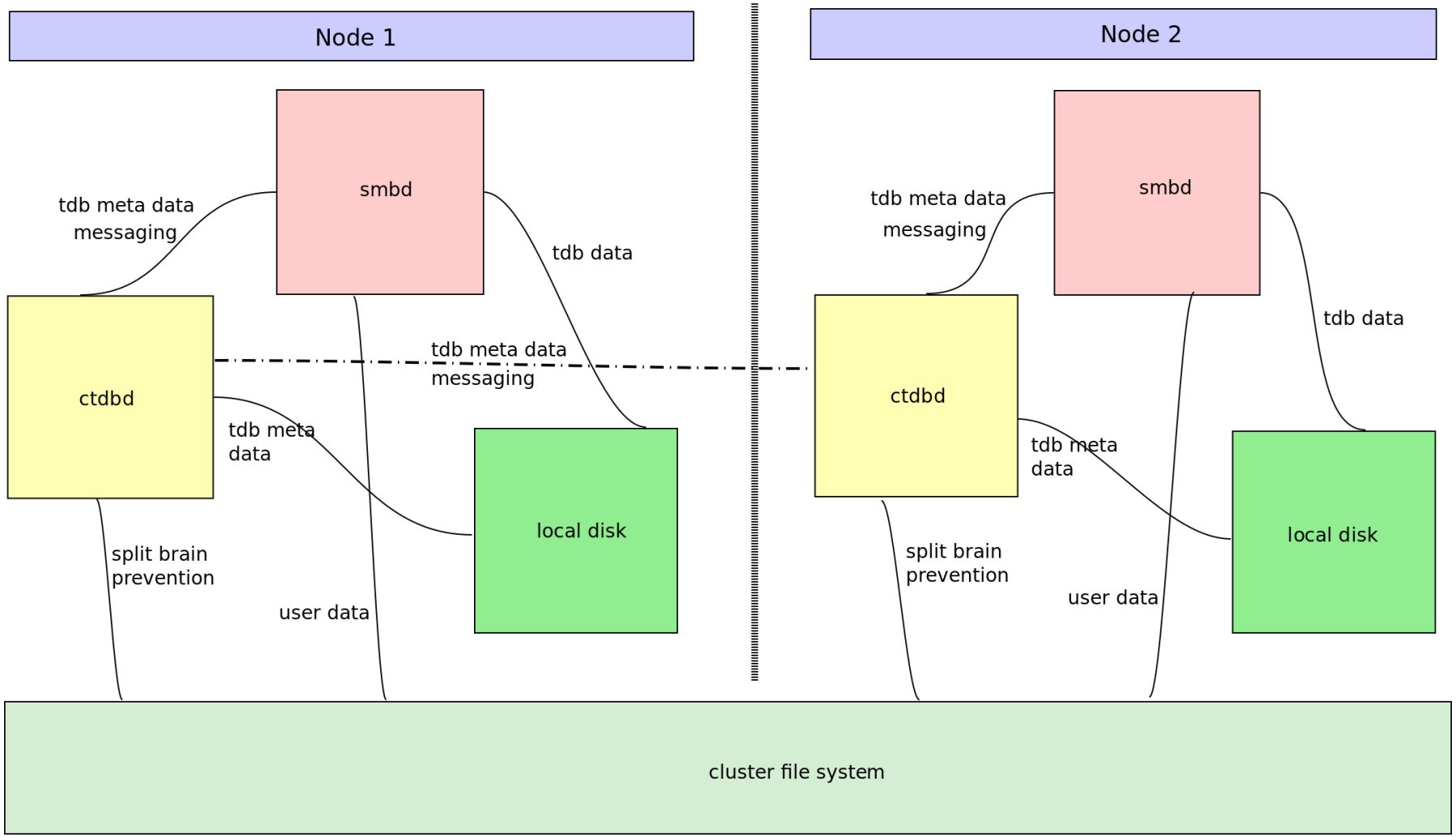
CTDB Design – General

- one daemon `ctdbd` on each node (and temporary forks)
- `smbd` talks to local ctdbd for messaging and TDB access
- `ctdbd` handles metadata of TDBs via the network
- `ctdbd` keeps local TDB copy (LTDB) for fast data reads/writes
- the actual record read and write ops are directly to the LTDB
- normal and persistent TDBs are handled differently
- Note: each single samba client connection is served by a single node and *not* spread across multiple nodes.

CTDB Design – General

- one daemon `ctdbd` on each node (and temporary forks)
- `smbd` talks to local ctdbd for messaging and TDB access
- `ctdbd` handles metadata of TDBs via the network
- `ctdbd` keeps local TDB copy (LTDB) for fast data reads/writes
- the actual record read and write ops are directly to the LTDB
- normal and persistent TDBs are handled differently
- Note: each single samba client connection is served by a single node and *not* spread across multiple nodes.

CTDB Design – Daemons



CTDB Design – Management

- HA and cluster management features:
- monitor health of ctdb cluster and (optionally) services (Samba, NFS, ...)
- fail over/fail back IP addresses and services
- In case of failover, connections are lost: clients need to reconnect ($\Rightarrow$ tickle acks)
- *Recovery* ensures consistent state of DBs in case of failure

CTDB Design – Management

- HA and cluster management features:
- monitor health of ctdb cluster and (optionally) services (Samba, NFS, ...)
- fail over/fail back IP addresses and services
- In case of failover, connections are lost: clients need to reconnect ($\Rightarrow$ tickle acks)
- *Recovery* ensures consistent state of DBs in case of failure

CTDB Design – Management

- HA and cluster management features:
- monitor health of ctdb cluster and (optionally) services (Samba, NFS, . . .)
- fail over/fail back IP addresses and services
- In case of failover, connections are lost: clients need to reconnect ($\Rightarrow$ tickle acks)
- *Recovery* ensures consistent state of DBs in case of failure

CTDB Design – Management

- HA and cluster management features:
- monitor health of ctdb cluster and (optionally) services (Samba, NFS, . . .)
- fail over/fail back IP addresses and services
- In case of failover, connections are lost: clients need to reconnect ($\Rightarrow$ tickle acks)
- *Recovery* ensures consistent state of DBs in case of failure

CTDB Design – Management

- HA and cluster management features:
- monitor health of ctdb cluster and (optionally) services (Samba, NFS, . . .)
- fail over/fail back IP addresses and services
- In case of failover, connections are lost: clients need to reconnect ($\Rightarrow$ tickle acks)
- *Recovery* ensures consistent state of DBs in case of failure

CTDB Design – Node Roles

- *recovery master* of the cluster (recmaster)
- *data master* of a record (DMASTER)
- *location master* of a record (LMASTER)

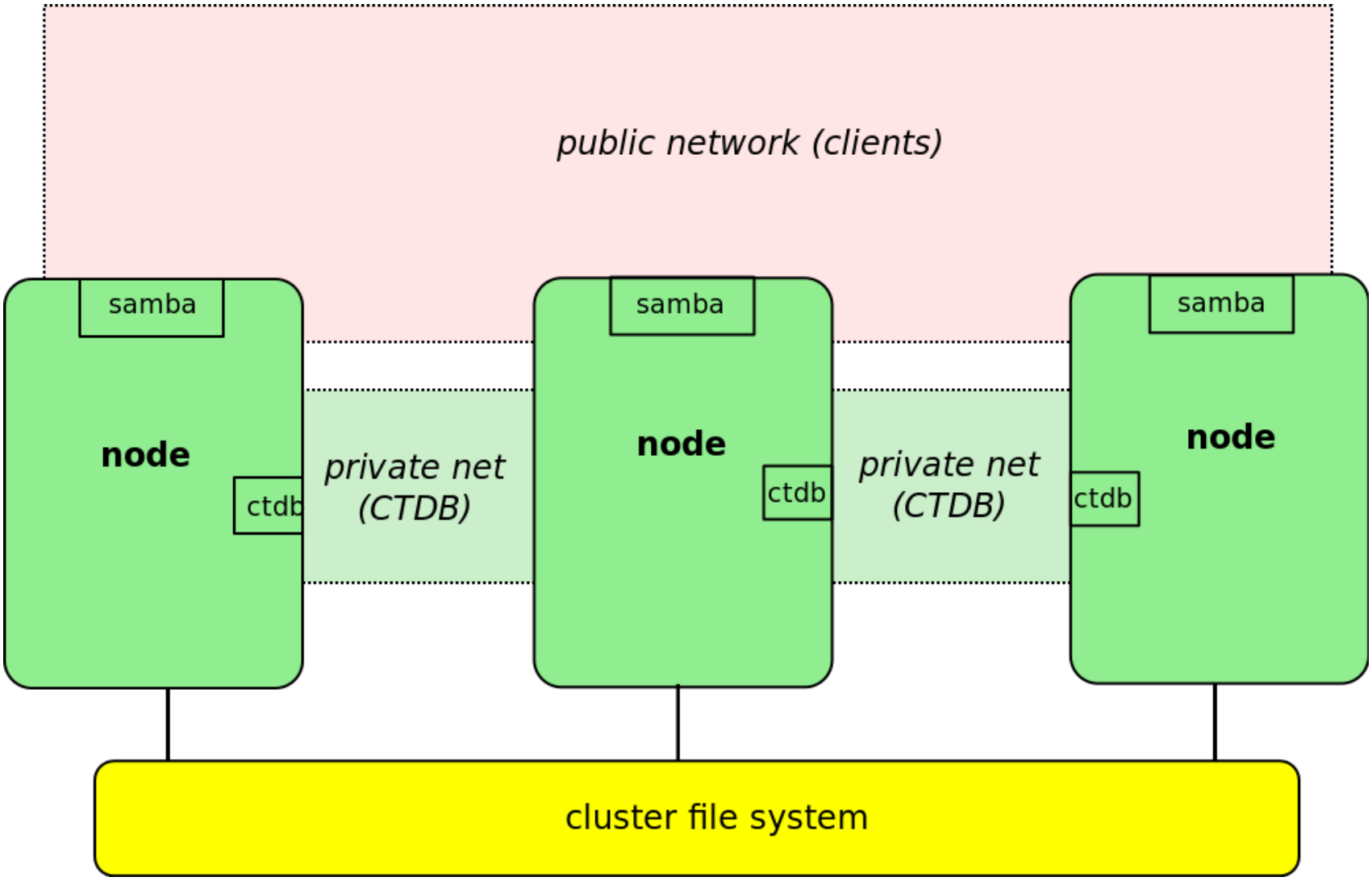
CTDB Design – Node Roles

- *recovery master* of the cluster (recmaster)
- *data master* of a record (DMASTER)
- *location master* of a record (LMASTER)

CTDB Design – Node Roles

- *recovery master* of the cluster (recmaster)
- *data master* of a record (DMASTER)
- *location master* of a record (LMASTER)

CTDB - Basic Setup



CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Configuration

- central file: `/etc/sysconfig/ctdb`
- debian based: `/etc/default/ctdb`
- set `CTDB_RECOVERY_LOCK` for split brain prevention
- fill `/etc/ctdb/nodes` with internal node addresses (one address per node)
- same file on all nodes!
- set `CTDB_PUBLIC_ADDRESSES` in `/etc/sysconfig/ctdb`
- typical value `/etc/ctdb/public_addresses`
- lines of the form `address/netmask interface`
- pool of addresses that ctdb can distribute across the cluster
- consider `CTDB_SYSLOG=yes` (logrotation missing...)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

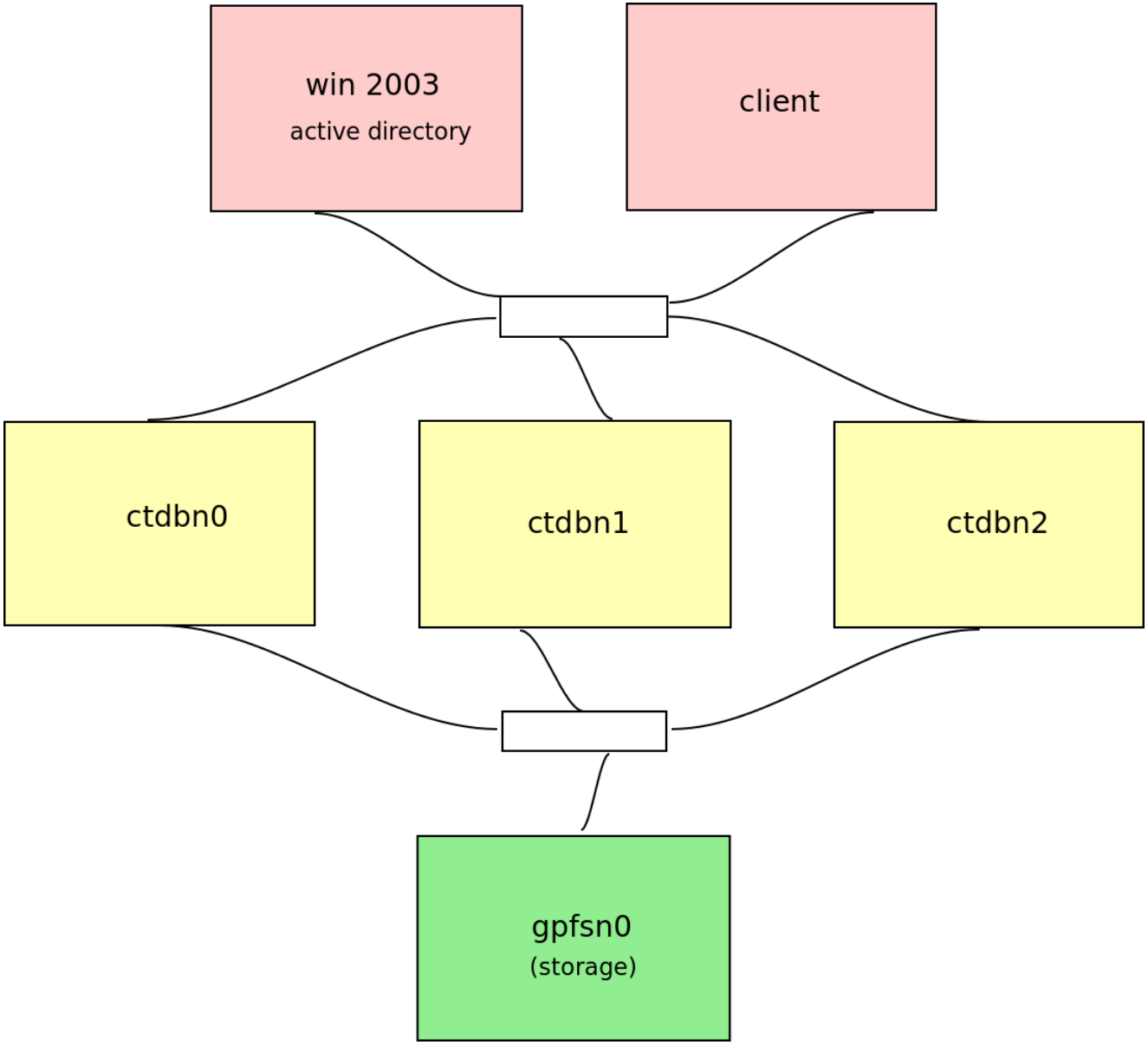
CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

CTDB - Toolbox

- `ping_pong`: test the cluster file system POSIX locks
- `ctdbd`: ctdb daemon
- `ctdb`: commandline tool to query/control `ctdbd` and manage databases
  - `ctdb status`
  - `ctdb ip`
  - `ctdb scriptstatus`
  - `ctdb getdbmap`
  - `ctdb catdb`
  - `ctdb backupdb` / `ctdb restoredb`
  - `ctdb wipedb`
- `onnode`: run commands on all or on selected nodes
- `ltdbtool`: query and convert local tdb copies (NEW!)

Demo Cluster Layout



Clustered Samba

- upstream since version 3.3
- transaction rewrite in 3.5.2
- cluster branches `v3-4-ctdb` and `v3-6-ctdb` in `git://git.samba.org/obnox/samba-ctdb.git`
- build: `configure --with-cluster-support`
- may: add `idmap_tdb2` to shared modules
- verify that `gpfs.so` is build for GPFS usage (hint: only need the open sourced headers for building)
- identical `smb.conf` on all nodes!
- hint: registry configuration

Clustered Samba

- upstream since version 3.3
- transaction rewrite in 3.5.2
- cluster branches `v3-4-ctdb` and `v3-6-ctdb` in `git://git.samba.org/obnox/samba-ctdb.git`
- build: `configure --with-cluster-support`
- may: add `idmap_tdb2` to shared modules
- verify that `gpfs.so` is build for GPFS usage (hint: only need the open sourced headers for building)
- identical `smb.conf` on all nodes!
- hint: registry configuration

Clustered Samba

- upstream since version 3.3
- transaction rewrite in 3.5.2
- cluster branches `v3-4-ctdb` and `v3-6-ctdb` in `git://git.samba.org/obnox/samba-ctdb.git`
- build: `configure --with-cluster-support`
- may: add `idmap_tdb2` to shared modules
- verify that `gpfs.so` is build for GPFS usage
  (hint: only need the open sourced headers for building)
- identical `smb.conf` on all nodes!
- hint: registry configuration

Clustered Samba

- upstream since version 3.3
- transaction rewrite in 3.5.2
- cluster branches `v3-4-ctdb` and `v3-6-ctdb` in `git://git.samba.org/obnox/samba-ctdb.git`
- build: `configure --with-cluster-support`
- may: add `idmap_tdb2` to shared modules
- verify that `gpfs.so` is build for GPFS usage (hint: only need the open sourced headers for building)
- identical `smb.conf` on all nodes!
- hint: registry configuration

Clustered Samba

- upstream since version 3.3
- transaction rewrite in 3.5.2
- cluster branches `v3-4-ctdb` and `v3-6-ctdb` in `git://git.samba.org/obnox/samba-ctdb.git`
- build: `configure --with-cluster-support`
- may: add `idmap_tdb2` to shared modules
- verify that `gpfs.so` is build for GPFS usage (hint: only need the open sourced headers for building)
- identical `smb.conf` on all nodes!
- hint: registry configuration

Clustered Samba

- upstream since version 3.3
- transaction rewrite in 3.5.2
- cluster branches `v3-4-ctdb` and `v3-6-ctdb` in `git://git.samba.org/obnox/samba-ctdb.git`
- build: `configure --with-cluster-support`
- may: add `idmap_tdb2` to shared modules
- verify that `gpfs.so` is build for GPFS usage (hint: only need the open sourced headers for building)
- identical `smb.conf` on all nodes!
- hint: registry configuration

Clustered Samba

- upstream since version 3.3
- transaction rewrite in 3.5.2
- cluster branches `v3-4-ctdb` and `v3-6-ctdb` in `git://git.samba.org/obnox/samba-ctdb.git`
- build: `configure --with-cluster-support`
- may: add `idmap_tdb2` to shared modules
- verify that `gpfs.so` is build for GPFS usage (hint: only need the open sourced headers for building)
- identical `smb.conf` on all nodes!
- hint: registry configuration

Clustered Samba

- upstream since version 3.3
- transaction rewrite in 3.5.2
- cluster branches `v3-4-ctdb` and `v3-6-ctdb` in `git://git.samba.org/obnox/samba-ctdb.git`
- build: `configure --with-cluster-support`
- may: add `idmap_tdb2` to shared modules
- verify that `gpfs.so` is build for GPFS usage (hint: only need the open sourced headers for building)
- identical `smb.conf` on all nodes!
- hint: registry configuration

smb.conf: basic example

```
[global]
    clustering = yes
    netbios name = smbcluster
    workgroup = sambaxp
    realm = sambaxp.private
    security = ads
    #passdb backend = tdbsam
    #groupdb:backend = tdb

    idmap backend = tdb
    idmap uid = 1000000-2000000
    idmap gid = 1000000-2000000

    fileid:algorithm = fsname

[share]
    path = /cluster_storage/share
    writeable = yes
    vfs objects = fileid
```

smb.conf: gpfs options

```
clustering = yes

fileid:algorithm = fsname

#gpfs:sharemodes = yes
#gpfs:leases = yes
gpfs:winattr = yes
#gpfs:getrealfilename = yes
#gpfs:refuse_dacl_protected = no

force unknown acl user = yes
nfs4: mode = special
nfs4: chown = yes
nfs4: acedup = merge

vfs objects = fileid gpfs syncops
```

Let's work with our cluster! ...

- internal cluster network: `172.22.18.0/22`
  - `gpfsn0 : 172.22.18.60`
  - `ctbdn0 : 172.22.18.80`
  - `ctbdn1 : 172.22.18.81`
  - `ctbdn2 : 172.22.18.82`
- external client network: `172.22.17.0/22`

Thank you very much!