Two years with Samba's autobuild

Samba

- Samba is much more than a file server these days

  - AD Domain Controller (my particular interest)

  - Classic (NT4-like) domain controller

  - Domain Member

    - Including in support of other applications like squid, or desktop logon

  - Print server

  - Client library (libsmbclient, libnetapi)

Samba is an old project

- Over 20 years old now

- Global contributor base

- Conservative

- Historically very portable

  - Unix was much more diverse 20 years ago

- Continuous development modal

  - HEAD/master/trunk

  - Current release branch

Samba really needs testing:

- Samba is:

  - Very Large

  - Extremely portable

  - Extremely complex

- Testing a key part of Network protocol analysis:

  - Make Smbtorture run against windows

  - Then run smbtorture against Samba to ensure we match

We always wanted to test

- But we did not always get around to it

- For many years we had trouble keeping the tree compiling!

- Increasing automation

  - Over 10 years

  - Ending up at automated enforcement

  - Moore's Law helped a lot

The Challenge

- As a network server doing real TCP/IP

- And a file server using root to impersonate users

- We insisted on:

  - Non-root

  - Non-network

  - Non-destructive

  - Unattended and Secure Testing

Faking it

- To meet our non-root requirement we have:

- Wrappers for

    - Sockets (socket_wrapper)

    - NSS (nss_wrapper)

    - system uid/gid (uid_wrapper)

    - FS Extended Attributes (VFS module)

    - owner, group, mode (VFS module)

    - POSIX ACLS (VFS module)

- Yes, we sometimes find we are testing in an unreal world...

The Journey

- Samba's build farm for compile testing

- Functionality testing on the build farm

- make test

- Complex test environments

- subunit

- autobuild

- enforced autobuild

Automation has never been enough

- 'make' should always have passed on any developer workstation

  - Yet folks still broke the build

- For a decade now, it has been possible to run 'make test'

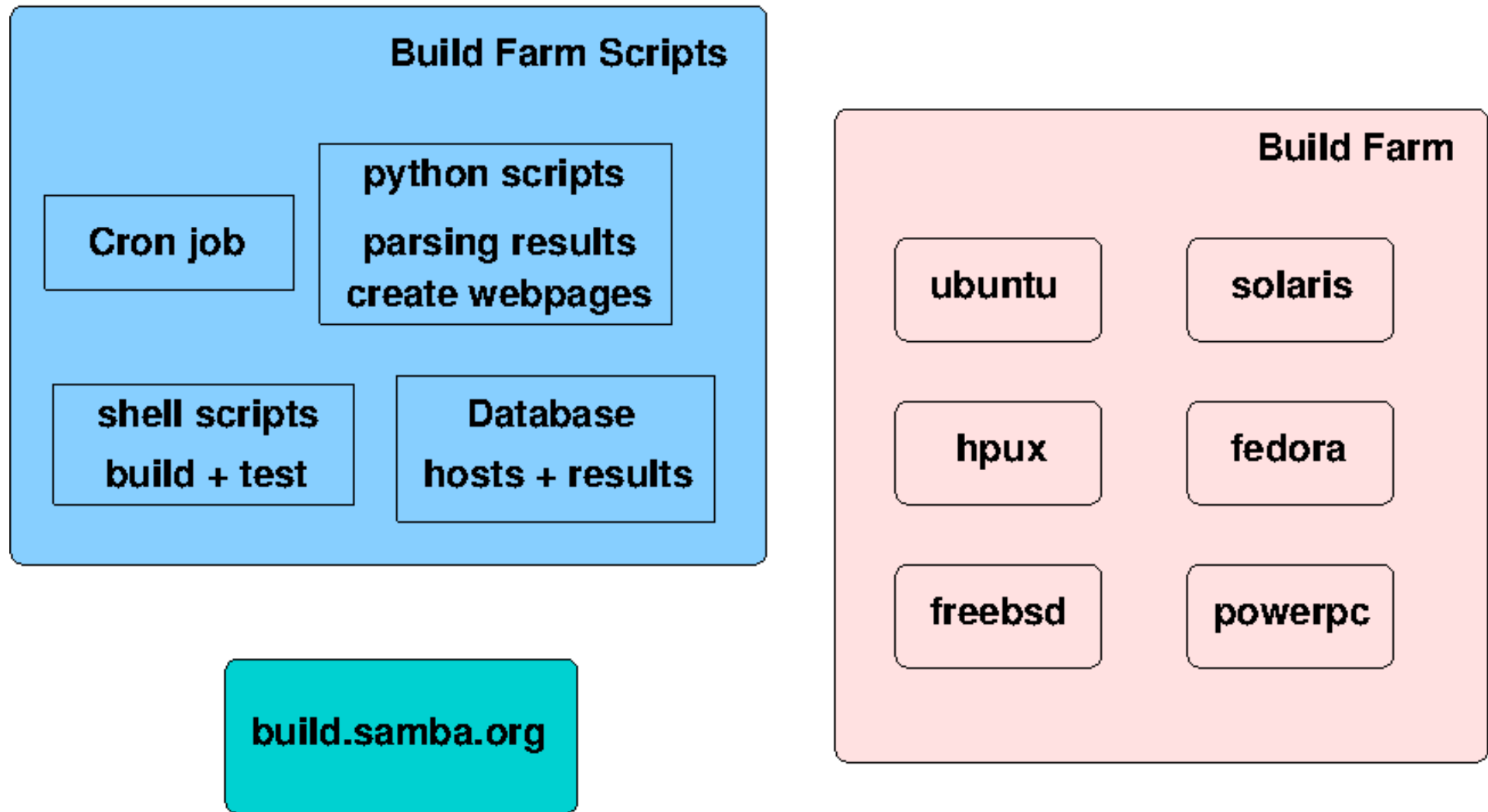 - Yet releases were made with make test failing

I could not have introduced that failure!

- The optimistic programmer

- Once 'make test' starts to fail...

  - Start only looking for catastrophic failures

- Once 'make test' takes too long

  - Push after most of the tests have run

- Our 'make test' currently takes 90min

The build farm isn't enough

- The build farm builds Samba on multiple platforms

- Typically donated cycles on hosted hardware

- Web page of output

- Reporting script e-mails 'guilty' developers

- Flaky tests

  - Developers started to ignore the mails
  - Little incentive to fix up after a commit is in

Build Farm

Build Farm Scripts

Cron job

python scripts

parsing results

create webpages

shell scripts

build + test

Database

hosts + results

build.samba.org

Build Farm

ubuntu

solaris

hpux

fedora

freebsd

powerpc

The rise of autobuild

- Started out of a team debate about code review

- Should the reviewer compile and test the code?

  - Even if that takes over an hour?

- What point is review if we don't even pass our own tests?

- Purchased a 24-core server

  - Hosted by SerNet in Germany

- Wrote autobuild.py

Samba does Not Invented Here Well

- Samba's CI system is simple but custom

- Python autobuild script

  - rebases git tree

  - Builds samba

  - Runs 'make test'

  - Installs samba

  - If successful, writes commits to git branch

Autobuild UI

- git autobuild

  - In .gitconfig:

    [alias]

    autobuild = push autobuild +HEAD:master

- autobuild-tail.sh

  - On sn-devel server

  - shows output as the test runs

- Mail to developer on failure

Autobuild enforcement

- First a trial period
    - Written September 2010
    - Incentive was that you could not accidentally break the tree
- Then social enforcement
    - Team policy as of October 2010
    - Only one breach of the social rule
- Eventually enforced by GIT hooks
    - March 2011
- Every master commit since Jan 2011 has been via autobuild

Autobuild hates me!

- Autobuild hates nobody

- But it can feel like it, if you run autobuild often

  - You may start running autobuild more often if you feel you have no chance to get a patch in

- Flaky tests may break your build

- Eventually marked in selftest/flapping

Getting to autobuild

- Autobuild is just the final wrapper

- The real work in in 'make test'

  - Multiple test frameworks

  - In multiple languages

  - All speak subunit or are converted to subunit

  - A mix of integration and unit testing

  - Long-lasting test environments due to startup cost

  - Complex test environment interactions

Many languages

```
make--
  python--sh-+-2*[python]
              |-sh--perl-+-perl-+-samba-+-13*[samba]
              |          |      |       `-samba--
2*[python]
              |          |      `-sh--tee
              |          `-sh-+-python
              |               `-test_samba_tool--
python
              `-2*[tee]
```

ABI check

- Public Library - published ABI (*.sigs)

- Changes in Library

  - Check if existing symbol removed (major)

  - Check if existing symbol definition changed (major)

  - Check if new symbol added (minor)

- Build time check

  - Developer doesn't have to worry

Dependency checks

- Duplicate Symbol Check

  - If any binary has two libraries which define the same symbol

  - Or define a symbol used in system library

- Circular Dependency Check

  - a depends on b, b depends on c, c depends on a

Two types of test

- Unit test

  - Tests standalone components

  - Easy to run

- Environment test

  - Needs samba server (environment) running

  - Socket wrappers to fake IP addresses

    - Difficult to set up and run manually

    - How to debug environment tests?

Test framework

- Perl-based test environment builder and test runner

- Testtools

  - Python unit testing framework

- Subunit

  - Streaming protocol for test results

  - C / C++ / Python / Perl / Shell

- Samba extension

  - subunit extension to arrange tests in suites

    – 9000+ tests arranged as 1298 testsuites

Writing tests (Python)

For env in ["fl2000dc", "fl2003dc"]:
    plantestsuite("samba4.ldap_schema.python(%s)" % env,
                    env,
                    "python dsdb/tests/python/ldap_schema.py $SERVER
                        -U$USERNAME%$PASSWORD --workgroup=$DOMAIN"   )

-- TEST --
samba4.ldap_schema.python(fl2000dc)
fl2000dc
python dsdb/tests/python/ldap_schema.py $SERVER -U"$USERNAME%$PASSWORD"
--workgroup=$DOMAIN 2>&1
    | selftest/filter-subunit --fail-on-empty --prefix="samba4.ldap_schema.python(fl2000dc)."

-- TEST --
samba4.ldap_schema.python(fl2003dc)
fl2003dc
python dsdb/tests/python/ldap_schema.py $SERVER -U"$USERNAME%$PASSWORD"
--workgroup=$DOMAIN 2>&1
    | selftest/filter-subunit --fail-on-empty --prefix="samba4.ldap_schema.python(fl2003dc)."

Writing tests (Shell)

plantestsuite("samba4.blackbox.samba_tool(dc:local)",
       "dc:local",
       "utils/tests/test_samba_tool.sh '$SERVER' '$SERVER_IP'
            '$USERNAME' '$PASSWORD' '$DOMAIN' ")

-------------------------------------- test_samba_tool.sh --------------------------------------

testit "Test login with --machine-pass without kerberos" $VALGRIND $smbclient -c 'ls'
     $CONFIGURATION //$SERVER/tmp --machine-pass -k no

testit "Test login with --machine-pass and kerberos" $VALGRIND $smbclient -c 'ls'
     $CONFIGURATION //$SERVER/tmp --machine-pass -k yes

testit "time" $VALGRIND $samba_tool time $SERVER $CONFIGURATION
     -W "$DOMAIN" -U"$USERNAME%$PASSWORD" $@

testit "domain level.show" $VALGRIND $samba_tool domain level show

testit "domain info" $VALGRIND $samba_tool domain info $SERVER_IP

Test environment debugging

- Vital to be able to debug a test as if the test was running

- How to set up an environment?

  - waf --testenv [SELFTEST_TESTENV=fl2003dc]

    - launches xterm (client)

- How to debug samba server?

  - waf --testenv --gdbtest

    - launches two xterms (gdb attached to server + client)

- What if debugging on a remote machine without X

  - waf --testenv --screen   (shell script wrapper)

    - Use screen program to start multiple sessions (environment + client)

Wintest: The holy grail?

- Automated testing against Windows

- Uses VM snapshots

- Runs really as root

- Binds to real ports

- Does things like a windows domain join, join Samba as a DC to a windows domain

Wintest: The good and the bad

- Sadly:

  - Difficult to set up and unreliable

  - Took a lot of effort to create

  - Never really taken up by the rest of the team (only 4 users ever worldwide)

- Thankfully:

  - Still saved our bacon on Samba 4.0

  - A late patch broke domain joins from Windows