Samba and the road to Python3

Noel Power
SUSE/Samba team
noel.power@suse.com
npower@samba.org
Agenda

• Reasons to move to Python3
• What is supported in what release
• Some history of the porting effort
• Challenges
• Lessons learned
• Questions
Why move to Python3

• Python 2 is quickly approaching its EOL and will no longer be supported upstream after 2020-01-01
• Enterprise vendors already backing away from Python2
• SUSE
  – Python 2 is going to be removed from Open Suse Factory on 2020-01-02. Python2 will not be supported in SLE15-sp1, Python3 is the default
• Red Hat
  – RHEL 8, Python 3 is the default, Python2 not supported
• Python2 suffers from some genuine problems that make moving to Python3 compelling in it's own right
• In the end really we have no choice, the only choices here were how, when and how soon.
Usage of python in Samba

• Build system is written in python
• Approximately 70 c-python modules (used mostly by samba AD)
• Parts of the test environment are written in python
• Many unit tests are written in python
• Some non python tests call functionality written in python
What is supported in what release

• Samba 4.9
  – rudimentary support for python3 with --extra-python, this
  – can build the c-python modules against python3
  – can run some python test scripts under python3
  – supports Python2.6

• Samba 4.10
  – code fully supports python3, code is python2 *AND* python3 compatible
  – possible to build with either python3 (default) *OR* python2 (for legacy)
    • PYTHON=python2 ./configure && PYTHON=python2 make
  – python3 build can build with --extra-python=python2 (building c-python modules for both python2 & python3)
  – supports Python2.6+
  – supports Python3.4+
What is supported in what release

• Samba 4.11
  – supports Python2.6+ only in the limited sense that the build will work with
    • PYTHON=python2 ./configure --disable-python
• currently supports Python3.4+
• Future Releases
  – bump supported Python3 version ?
  – remove some remnants of python2/python3 scaffolding
    • at least remove c compatibility layer code and rewrite
    • remove python compatibility layer code and rewrite
Python3 in samba (some history)

• nearly 10 years ago python3 was first mentioned
  commit c24240bcd2f833321f45ea4ce0b6c6d080a3b990
  Author: Andrew Tridgell <tridge@samba.org>
  Date:   Wed Oct 6 20:11:01 2010 +1100

  waf: fixed some python3.x portability issues

  these have crept into the tree over time. Maybe we should add testing
  of a range of python versions to autobuild?

• A couple more of misc commits after that but really nothing more happened till
  sometime around start of 2015 when there seemed a bit serious effort to not so
  much port to python3 but put some infrastructure in place where that work could
  be started.
Python3 in samba (some history)

• A key commit here properly marked key foundations for python3 support
  commit 616dfae8ffa88bd6b8b1145bd9d75c5b873e7044
  Author: Petr Viktorin <pviktori@redhat.com>
  Date:    Thu Jan 15 14:22:22 2015 +0100

    buildtools: Add --extra-python configure option

    This allows building Python support for two different Python versions
    at the same time.

• And Petr followed up with cleaning up and improving various bits of the existing
  python c-api in pyldb, pytalloc etc.

• porting pyldb, pytalloc etc. to python3
  – Jan 2015 port pytalloc
  – May 2015 port pytevent and pytdb
  – Jun 2015 port pyldb
Python3 in samba (some history)

• Dec 2016 – Jan 2018 Lumir Balhar (Redhat) ported many of the c-python modules
• Jun 2018 – Waf 2.0.8 (Alexander)
• March 2019 4.10 released (first release fully supporting python3)
Samba Python porting challenges

- knowledge
  - samba is a pretty complex code base
  - c python modules require some core samba knowledge in order to decide what changes are necessary for python3 and even more knowledge to test those changes.
  - same can be said for the python code (requires some Samba AD specific experience)
  - python2/python3 knowledge
  - Build system (in python) based on WAF is not well understood is quite customized for samba
Samba Python porting challenges

• nature of the code base suggested a 'transition' to python3
  – samba project is a mixture of perl, shell, python & c
  – throwing away python2 and moving to python3 while attractive just practically wasn't possible for various reasons
    • risk
    • need to give users and vendors time to transition (at least a release)
    • would have meant effectively maintaining 2 separate code streams, making sure they functionally were kept in sync until time for switching.
  
• Porting code code to be Python2/Python3 compatible is challenging
• Making substantial code changes to existing code base is also risky
Python2/Python3 differences

• syntax changes
  – e.g. octal literals ‘0720’ → ‘0o7200’
  – Backticks removed e.g. `var` → ‘repr(var)’
  – Exceptions:
    'exception ValueError, e:'
    vs
    'except ValueError as e:'
    'except SomeException, (num, msg):'
    vs
    'except ldb.LdbError as e:
      (num, msg) = e.args'
Python2/Python3 differences

• ‘print’ is now a function and not a statement e.g. ‘print foo’ → ‘print(foo)’
• long renamed to int (and has the potential to cause some issues where the previous limits of int were relevant)
• api changes (many, many, many)
  – zip, map and filter now return iterators instead of lists (this can cause some amazingly weird problems)
• Integer division now provides a float result (need to use ‘//’ floor division instead)
Python2/Python3 differences

• Text model
  – Python2
    • has 'str' type and no 'byte' type
    • has unicode type
    • print(type(b'123'[0])) yields <type 'str'>
    • chr() which returns a character whose ascii code is in the range range [0..255]
  – Python3
    • has 'str' type (which is equivalent to unicode type in python2)
    • has 'byte' type
    • has 'byte' array type
    • print(type(b'123'[0])) yields <class 'int'>
    • chr() which returns a string representing a character whose Unicode code point is the integer i range [0..1,114,111]
Methodology

• Initially ported some bulk changes necessary for python2/python3 compatibility
  – Verify that at least these changes didn’t break python2 with CI
• Ported python based tests piecemeal to run with python2 & python3
  – Aimed for manageable byte sized changes that could be reviewed and tested
  – Maintained flexibility to enable larger changes to be applied when necessary e.g. sometimes importing just a single module in a test running in python3 could precipitate a ripple of changes.
  – Verification of changes using CI
    • CI jobs modified to run independent python3 & python2 tests
• Parallel porting to Waf 2.0.8
• Subsequent port build itself to be python2/python3 compatible
  – Initially introducing some Python3 specific CI jobs (just building)
  – Finally transitioning of CI jobs from being default built with python2 to default built with python3 (and swapping the python2/python3 CI jobs)
Some Statistics

• Multi-year effort 2015-2019
• Some trawling of the git repository with some very much finger in the air estimation
  ~1400 files changed
  ~42000 lines inserted
  ~11400 lines deleted
Lessons learned

• Moving to a python2/python3 compatible code base is a sensible transition step
• Be flexible
• Post releasing a working release that is python2/python3 compatible it is imperative to quickly move to removing python2 support completely
• CI is mandatory !!!! (gitlab CI and/or similar really is your friend)
• You can never have enough unit tests
• Porting a complex project is extremely draining :-)

Special thanks To

- I’ve probably missed out someone (sorry) but the main people to thank for the python3 porting are
  - Alexander Bokovoy
  - Andrew Bartlet
  - Douglas Bagnall
  - Joe Guo
  - Lumir Balhar
  - Petr Viktorin
Useful links

• The Conservative Python 3 Porting Guide

• Cheat Sheet: Writing Python 2-3 compatible code
  – https://python-future.org/compatible_idioms.html

• Porting Python 2 Code to Python 3
  – https://docs.python.org/3/howto/pyporting.html