NCLU Testing

Why do we test?

- Verify: we don’t break things, new features
- Dev cycle
  - *should* increase productivity: give developer quick feedback and identify errors earlier in the product development life cycle
- Design/de-coupled code
  - Example: separate implementation details of components so parts can be mocked without affecting test value
  - Example: functions/classes should have reduced scope which is tested individually
  - Counter:
    - Do not allow testing to drive design: have balance
- Documentation
  - The tests are using your code, how does it look to be a user of your code?
- Value of tests affected by
  - bug prevention, dev cycle time, speed, complexity, predictability

Integration tests

- Before we merge dev-next -> dev, full nclu-smoke tests *should* be run
- Full test run can take 6-10 hrs
- Failures need to be inspected one-by-one
- TRDB: tests run according to different schedules: often you'll look up results for a test and it'll only have been run once in the last 10 days. So it doesn't help much at all.
- Sometimes tests lack granularity and proper reporting--can be difficult to understand where the failure is.

Current NCLU dev cycle

- “dev-next” branch
- Run integration tests
- Success!
  - Merge to “dev”
- Failure :(
  - Diagnose, re-run, fix, hope things are still okay

Current bug fix cycle

- Issue assigned
- Reproduce on switch
- Modify capturing.py
- Put pdb somewhere
- Stop netd
- Start netd in foreground
- Modify NCLU python files with fix
  - Either directly or by dev local, building deb and installing new deb
- No integration test to verify fix
- No unit testing in dev cycle

“Unit” testing

- Less about “unit” testing
  - As you’ll notice, a lot of the “unit” tests actually feel like mocked integration tests
  - Coverage with implementation tested is more important than “unit” testing
- More about testing the code directly
- Instead of needing to run integration tests

Unit test dev cycle(proposal)

- Use branches for bugs and features
- Have 95%+ unit test coverage
- Run unit tests on every branch every time a commit is made
- If tests pass and code is approved, merge to dev-next
- If nightly dev-next tests pass, merge to dev automatically
- (debate on if dev-next is necessary)

Unit test bug fix cycle(proposal)

- Issue assigned
- Reproduce on switch
- Reproduce with test failure
- Fix, test passes now, yay!
- Test on switch and verify fix
- (Still no integration test to verify fix)
- Results:
  - Faster development time
  - Identify regressions earlier
  - Bootstrapping errors are identified immediately
  - Runtime errors can also often be identified

Current NCLU unit test compromises

- System dependencies
  - File system
  - Commands
- Import time side-effects
  - Makes patching very difficult
  - Imports need to be in test methods to defer import time side-effects to be after patches are done
- Python 2.7 base unit test package
- No setuptools

Unit test patches

- Patches
  - all file operations
  - all system commands
  - frr-reload
  - apt
- One global patch object
- setUp/tearDown on the global patch object for each test

Unit testing NCLU

- nclu.tests.base.BaseTest: patcher setUp/tearDown
- self.patcher.expect_command
- self.patcher.expect_file

Test runner

- (/home/nathan/test-runner): Should I move it to a repo somewhere?
- Run tests against tags and branches
- Updates hard node
- Build nclu deb
- Install deb
- Test reporting/coverage
- Fabric file

Improvements

- Make patcher work as context manager
- Patcher should cause test failures if expected commands *not* run(we do this in the experimental branch for example)
- Stacked patched file system
  - Layer groups of file configurations
  - Be able to quickly revert back to previous states in stack
- Test runner could send email alerts on:
  - New failures
  - Fabfile errors when trying to run

Bonus: dev cycle with VM

- If running tests, you can not develop against hard node
- Virtual machine is nice in-between
- Sync current dev nclu to VM switch automatically

```
rsync --update -v -r -e ssh nclu/nclu/ -e "ssh -p 22222"
cumulus@localhost:/usr/lib/python2.7/dist-packages/nclu/

while inotifywait -r nclu/nclu*; do

    rsync --update -v -r -e ssh nclu/nclu/ -e "ssh -p 22222"
cumulus@localhost:/usr/lib/python2.7/dist-packages/nclu/

done
```