Python Migration

Eliminate src/externals

1. Use available RPM packages first. For example: Use python-flask as it is available in the standard repo.
2. For Pegasus we currently do `python setup.py clean --all install_lib -d ${dist.python-lib}`, which is equivalent to `pip install --prefix <dir> --no-deps .`
3. For packages not available as RPM, like python-pika, we could take one of the following approaches,
   a. We could convert pika into RPM/DEB using snippet below
   b. pip install --prefix <dir> --no-deps pika # It gets installed in the same manner as Pegasus.
   c. Install using pip in RPM's pre/post install hooks
4. For binary tarballs, we would just execute `pip install --prefix <dir> <src>` <src> means lib/pegasus/python. All Python dependencies will be installed from PyPi.

We should no longer need src/externals, and thus not need `pegasus-config --python-externals`. For RPM and DEB packages setting PYTHONPATH would not be required at all. For binary tarballs we could ship setup.sh, like HTCondor that can set PYTHONPATH with simply `pegasus-config --python`.

Pros

1. Eliminates manual management of python dependencies in src/externals
2. Eliminates need to set PYTHONPATH for RPM/DEB packages etc.
3. Easy to add new packages.

RPM/DEB From PyPi

```
# https://stackoverflow.com/a/48285243

cd pegasus-src/lib/pegasus/python/

# Debian 9
docker run -i --rm -w /srv -v `pwd`:/srv debian:9 bash  -s <<EOT
apt-get update && apt-get -y install python-pip python-setuptools debhelper
pip install stdeb
rm -rf ./x
pip download --no-binary=:all: . -d x
cd x
tar zxvf pika-1.1.0.tar.gz
cd pika-1.1.0
python setup.py --command-packages=stdeb.command bdist_deb
EOT

# CentOS 7
docker run -i --rm -w /srv -v `pwd`:/srv centos:7 bash  -s <<EOT
yum -y install epel-release
yum -y install python36-pip python36-setuptools rpm-build
rm -rf ./x
pip download --no-binary=:all: . -d x
cd x
tar zxvf pika-1.1.0.tar.gz
cd pika-1.1.0
python3 setup.py bdist --formats=rpm
```

Testing

Using pytest instead of unittest

Uses assert instead of self.assert*.

```
import pytest

# Simple method, no need for classes
def test_method():
  # Simple assert, no need for assertEqual, etc.
  assert a % 2 == 0, "value was odd, should be even"

  with pytest.raises(ZeroDivisionError):
        1 / 0

# One method, multiple tests
# https://docs.pytest.org/en/latest/parametrize.html
@pytest.mark.parametrize(
    "a, b",
    [
                (1, 2, 3),
                (2, 3, 5),
                (5, -100, -95),
        ],
)
def test_eval(a, b, expected):
    assert a + b == expected

# Dependency Injection
@pytest.fixture(scope="function OR class OR module OR package OR session")
def client():
        import requests
         s = requests.Session()
        s.get(".../login")
        yield return s # Yield can be replaced with return if no cleanup is required.
        s.get(".../logout")
    s.close()

def test_external(client):
  assert client.get("/endpoint-1")
```

See: https://docs.pytest.org/en/latest/contents.html

Pegasus Code except Worker Package

Continue src/externals

👎

Platform specific packages, python-flask

Will need to test with the lowest common available version across supported platforms.

Will require more platform specific integration tests.

Run pre/post install scripts to install using pip, i.e. pip install Flask.

Mats Rynge Do .rpm, .deb packages support this? Mats: this is only allowed if you do into your own space (which is similar to what we do with externals. You are not allowed to pip install to the system location)

Packages may conflict with system installed packages. yum install python-flask (v0.9) vs pip install Flask (v1.0.2)

> *Latter two will require more platform specific integration tests to detect incompatible, missing dependencies.*

Worker Package

Continue src/externals

Only for packages not expected to be available on supported platforms.

"Vendorize"

Copy dependency code into Pegasus.vendor directory.

```
# Instead of this
import six

# Use this
import Pegasus.vendor.six
```

-- RESOLVED --

Minimum Python Version

Pegasus Code except Worker Package

Lowest common version across supported platforms is

Worker Package

Lowest common version across supported platforms is /

Mats Rynge Can we expect Python 2.7 to be available on CentOS 6?

Mats Rynge List Python components included in worker package.

1. Use python-six compatibility library.
2. Refactor code such that Python 3 code is made Python 2 compatible instead of the opposite.
3. See: https://python-future.org/compatible_idioms.html
4. See: https://docs.python.org/3/howto/pyporting.html