Sustainable software packaging for end users with conda

Chris Burr, Henry Schreiner, Enrico Guiraud, Javier Cervantes

CHEP 2019 © 5th November 2019
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.
Self compiled against Python 3.6
(but broken since updating to 3.7 homebrew)

MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.
conda create --name my-analysis \
  python=3.7 ipython pandas matplotlib \
  root boost  \
  tensorflow xgboost \
  bash top vim git
➢ Language agnostic package manager (Python, C++, R, Julia, Rust, Go, Java, Ruby, Fortran, ...)

➢ Multi platform (Linux, macOS, Windows)

➢ Multi architecture (i386, x86_64, aarch64, ppc64le)

➢ Provides “environments” which are self contained sysroots in a folder
  ➢ No admin privileges required
  ➢ Easy to preserve long term

➢ Easily switch between Python versions, compilers and other packages

➢ Anaconda, Inc. provides around 2,000 packages
Language agnostic package manager (Python, C++, R, Julia, Rust, Go, Java, Ruby, Fortran, …)

Multi platform (Linux, macOS, Windows)

Multi architecture (i386, x86_64, aarch64, ppc64le)

Provides “environments” which are self contained sysroots in a folder
  No admin privileges required
  Easy to preserve long term

Easily switch between Python versions, compilers and other packages

Anaconda, Inc. provides around 2,000 packages
- Community maintained collection of conda packages
- Over 8,000 packages available and rapidly growing
- Over 1,600 maintainers
- Fiscally sponsored project of NumFOCUS
- Includes everything user facing (vim/curl/findutils/htop/…)

conda-forge

- Community maintained collection of conda packages
- Over 8,000 packages available and rapidly growing
- Over 1,600 maintainers
- Fiscally sponsored project of NumFOCUS
- Includes everything user facing (vim/curl/findutils/htop/…)
➤ pip is the standard way to install Python packages

➤ Should be as simple as

```
  pip install PACKAGE_NAME
```

➤ Conda and pip are each aware of packages installed by the other
  ➤ Without any unexpected side effects

➤ Less helpful once leaving the Python ecosystem
How does conda-forge work?
Create a pull request against
https://github.com/conda-forge/staged-recipes

Can be mostly automated using
conda skeleton pypi zfit
 Updates

➤ Bots monitor for new releases
➤ Even works with non-standard URLs
➤ Maintainers normally just have to click merge

It is very likely that the current package version for this Notes for merging this PR:

1. Feel free to push to the bot's branch to update this
2. The bot will almost always only open one PR per ve
Checklist before merging this PR:
- Dependencies have been updated if changed
- Tests have passed
- Updated license if changed and license_file is p
Note that the bot will stop issuing PRs if more than 3 Ve
open. If you don't want to package a particular version
If this PR was opened in error or needs to be updated p
The bot will close this PR and schedule another one.

Bots monitor for new releases
Even works with non-standard URLs
Maintainers normally just have to click merge
Conda only supports installing binaries*

Relies on the solver knowing about API/ABI compatibility

Packages with shared libraries should specify what their ABI stability is

Doesn’t necessarily restrict what you can do

- Variants can be used to provide a matrix of different builds
- BLAS can be provided by netlib, mkl, blis and openblas
- Several MPI variants
- TensorFlow has CPU and (several) GPU variants

*Some organisations mirror the conda-forge build infrastructure for their own internal use
What about when ABIs change? More 😁!

A line is added to a git repository

Pull requests appear that rebuild packages in the correct order

Current Migrations:

![Migration Status](attachment:image.png)

- done (3)
- in-pr (7)
- awaiting-pr (0)
- awaiting-parents (1)
- bot-error (0)
Migration is currently ongoing for ppc64le and aarch64 support
- ROOT is included as a target

Support for compiling CUDA with nvcc is rapidly maturing
- Adds three additional additional targets (different driver versions)
- Close to being fully supported by the conda-forge tooling
- GPU variants of packages are already being added
➤ Creates a relocatable self extracting archive of a conda environment

```
jcris computer_one $ source activate example
(example) jcris computer_one $ # Package the current environment
(example) jcris computer_one $ conda-pack
Collecting packages...
Packing environment at '/Users/jcris/anaconda/envs/example' to 'example'
[==================================] 100% Complete
(example) jcris computer_one $ ls
example.tar.gz
(example) jcris computer_one $ # The environment is packaged
(example) jcris computer_one $ # Get the file size
(example) jcris computer_one $ du -h example.tar.gz
55M   example.tar.gz
```

```
jcris computer_two $ # Unpack the environment
jcris computer_two $ mkdir myenv
jcris computer_two $ tar -xf example.tar.gz -C myenv
jcris computer_two $ ls
example.tar.gz   myenv
jcris computer_two $ # Activate the environment
jcris computer_two $ source myenv/bin/activate
(myenv) jcris computer_two $ # Use applications in the environment
(myenv) jcris computer_two $ which ipython
/Users/jcris/computer_two/myenv/bin/ipython
(myenv) jcris computer_two $ ipython
Python 3.6.5 |Anaconda, Inc.| (default, Apr 26 2018, 08:42:37)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.
```

Potential replacement for mock in DIRACOS?
https://indico.cern.ch/event/773049/contributions/3473353/

https://conda.github.io/conda-pack/
> Wheels are a way to distribute binaries that are installed using pip

> Create Python wheels from conda packages

```
$ conda press --subdir osx-64 --skip-python --fatten iminuit=1.3.7=py37h86efe34_0
created fat wheel: iminuit-1.3.7-0_py37h86efe34-cp37-cp37m-macosx_10_9_x86_64.whl
```

> Why?
  > Centralises the building of packages
  > Easier to pull in dependencies
  > Use newer ABIs than manylinux

➤ Conda-forge now includes:

- ROOT
- uproot
- Awkward Array
- zfit
- Pythia 8
- VOMS
- HepMC
- iminuit
- XRootD
- cppyy
- mcerp
- AlphaTwirl
- Boost histogram
- GEANT4
- RapidSim
- Cling
Reliably install ROOT in under 5 minutes on any machine (almost)
- Linux, macOS, and Windows Subsystem for Linux (cling doesn’t support native 64-bit Windows)

Complete installation with C++17, graphics, OpenGL

Seamlessly integrates with the rest of conda-forge
- No builtin dependencies (excluding cling for now)
- No PYTHONPATH/LD_LIBRARY_PATH mess
- Easily switch between versions of Python, ROOT and anything else

Downloaded over 50,000 times since it was released 9 months ago

Currently working with the ROOT team to integrate with their nightlies
- Plan to release the binaries to a dedicated conda channel
- If you’d be interested in using these, please let me know!
Having the ability to manage arbitrary environments is convenient

Sometimes more resources are needed (clusters or grid)
  - Running `conda install` on thousands of nodes is a bad idea

Solution: Deploy to CVMFS like other HEP software
  - Analysts make a pull request to add an `environment.yml` file to a git repository
  - Automatically deployed GitLab CI
  - An application of the Poster: “A gateway between Gitlab CI and DIRAC”
Conda allows end users to easily manage their own software environments

The quickest and most reliable way to install ROOT on arbitrary machines
1a) Download Linux and Windows Subsystem for Linux*

```bash
wget -nv http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh -O miniconda.sh
```

1b) Download macOS

```bash
wget -nv http://repo.continuum.io/miniconda/Miniconda3-latest-MacOSX-x86_64.sh -O miniconda.sh
```

2) Install conda

```bash
bash miniconda.sh -b -p $HOME/miniconda
source $HOME/miniconda/etc/profile.d/conda.sh
conda config --add channels conda-forge
```

3) Create an environment and activate it

```bash
conda create --name my-environment python=3.7 ipython root jupyterlab
conda activate my-environment
```

*WSL only as ROOT doesn’t support native 64-bit Windows
Questions?
How not to use pip

➤ Installing should be as simple as:

```
pip install PACKAGE_NAME
```

➤ Don’t use things that modify global state:

```
sudo pip install PACKAGE_NAME
```

➤ Interacts poorly with system package managers
➤ Can make it impossible to update or install packages using apt/yum/pacman/…

```
pip install --user PACKAGE_NAME
```

➤ Normally has a higher priority in the Python search order
➤ Can break other installations (e.g. use on lxplus can break your experiments software stack)

➤ venv allow you to create environments from arbitrary Python installs
➤ One repository per package ("feedstock")

➤ All packages are built using well known CI providers

➤ A year ago this was Travis CI + Circle CI + Appveyor

➤ Now mostly Azure Pipelines

➤ Native builds for alternative architectures:
  ➤ ppc64le with Travis CI
  ➤ aarch64 with Drone CI

➤ All managed by an external package: conda-smithy
  ➤ Used to regenerate CI configuration for each update
Long term reproducibility is important for science

Easy to export exact builds of everything that was installed

```
conda list --explicit --md5 > environment.txt
```

```
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: osx-64
@EXPLICIT
https://repo.anaconda.com/pkgs/main/osx-64/blas-1.0-openblas.conda#4ff605b9a6c88bbfd4428e6f9703d9ff
https://conda.anaconda.org/conda-forge/osx-64/bzip2-1.0.8-h01d97ff_1.tar.bz2#8397e58ec04ed7961aca25e741fa175a
https://conda.anaconda.org/conda-forge/osx-64/ca-certificates-2019.9.11-hedca14488_0.tar.bz2#2dd837660ef2c2b7c5535ec1a09b
https://conda.anaconda.org/conda-forge/noarch/conda-forge-pinning-2019_10_01_0.tar.bz2#97190c790b5b9e8e25bce2d150b0471
https://conda.anaconda.org/conda-forge/osx-64/ffiidi-1.0.5-h01d97ff_1002.tar.bz2#d5db22ff1a37c0f85f50099facb755c
https://conda.anaconda.org/conda-forge/osx-64/glib-5.1.7-h01d97ff_1.tar.bz2#796aa4523450c9ab1b495c5c47b10c859
https://conda.anaconda.org/conda-forge/osx-64/jPEG-9c-h1de35cc_1001.tar.bz2#bbcc9abfeb1c26568e1ec450234512
https://conda.anaconda.org/conda-forge/osx-64/libcxxabi-9.0.0-0.tar.bz2#d3d2041d2ecf4d8e8e62c855b0bef
https://conda.anaconda.org/conda-forge/osx-64/libgfortran-3.0.1-0.tar.bz2#d69b2cc1c82503957007268ac3f3d912e
https://conda.anaconda.org/conda-forge/osx-64/libiconv-1.15-h01d97ff_1005.tar.bz2#ec331659b70e6656054bf1ca65ac82
https://conda.anaconda.org/conda-forge/osx-64/libodium-1.0.17-h01d97ff_0.tar.bz2#3d1aeeaa242c631c14e4b3b144660fa7
https://conda.anaconda.org/conda-forge/osx-64/libuv-1.32.0-h01d97ff_0.tar.bz2#fcbcb890e331687f5b118371d3073464
https://conda.anaconda.org/conda-forge/osx-64/lvm-openmp-9.0.0-h40edb58_0.tar.bz2#7d3c684e4f166b92a6d5e50e25942d9
https://conda.anaconda.org/conda-forge/osx-64/loz2-2.10-h1de35cc_1000.tar.bz2#9cc86637d9d6ec9005376f5c5b296e0
https://conda.anaconda.org/conda-forge/osx-64/pandoc-2.7.3-0.tar.bz2#9d7c8563365d3980e633227bc6cfa25
https://conda.anaconda.org/conda-forge/osx-64/perl-5.26.2-haece8f3_1006.tar.bz2#7c0c2da3e20d98b1045fe3e945d3dc
https://conda.anaconda.org/conda-forge/osx-64/pixman-0.38.0-h01d97ff_1003.tar.bz2#db28595ad96073e3aad622db0b36068
```

```
conda create --name my-new-env --file environment.txt
```
Bad metadata in old packages can “poison” the solver
- Mechanism exists for patching this but it’s fiddly

Current solution is to move packages to a “broken” channel
- Channel list must have “conda-forge/label/broken” appended
- URL also changes (for now)

Lots of benefits to having a docker container “just in case”
- Currently setting up a mirroring proxy for conda within LHCb
- Also deploying to CVMFS
- Ask me if you're interested
➤ First: Don’t vendor or have builtin dependencies

➤ Second: Seriously...don’t...

➤ Third: Make it easy to unvendor them

➤ Finally: At least make sure they’re contained
  ➤ Statically link
  ➤ Ensure symbols are hidden
  ➤ Don’t put vendored files in standard locations (e.g. shared libraries in $PREFIX/lib/)

How To Make Package Managers Cry: https://www.youtube.com/watch?v=NSemlYagjIU