Language agnostic (Python/R/C++/Julia)

Multi-platform: (Windows/macOS/Linux)

Multi-architecture: (i386/x86_64/POWER8/POWER9 and I think arm64 is coming))

Installs binaries (No install from source support)

Very popular in the academic and commercial “Data Science” community
  Many popular libraries list it as the recommended install method

Has changed a lot in the last two years
- Conda can be into any location in three lines: (similar for macOS, GUI installer for windows)
  
  ```
  3  wget http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
  4  sh Miniconda3-latest-Linux-x86_64.sh "$HOME/miniconda"
  5  # Add to your shell .rc file:
  6  source "$HOME/miniconda/etc/profile.d/conda.sh" ....... # For bash-like shells
  7  source "$HOME/miniconda/etc/profile.d/conda.csh" ....... # For csh-like shells
  8  source "$HOME/miniconda/etc/fish/conf.d/conda.fish" ....... # For fish
  ```

- This used to be done with `export PATH=$CONDA_DIR/bin:$PATH`
  - Now uses a function instead
    - Prevents bad interactions with other software
    - Allows conda environments to modify the shell environment
    - The “root” environment can be set up with `conda activate root`

- Use of the base “root” environment is discouraged
- A conda “environment” is just a folder in $CONDA_DIR/envs/
  - Switch between them using conda [de]activate $ENV_NAME
  - Activating sets PATH and sources scripts in $CONDA_DIR/etc/conda/activate.d

- Create an environment with conda create --name $ENV_NAME python=3.7 matplotlib pandas

- When activated packages are installed with conda install scikit-learn

- Behind the scenes:
  - Packages are kept in $CONDA_DIR/pkgs/
  - Both a tarball-like file and it’s extracted contents
  - Mixture of copying, hard links and symlinks to the environment’s directory
Sources of packages
The default source is the Anaconda Distribution

- Mostly Python/R packages for Data Science
  - Contains over 1,400 packages
  - Includes proprietary software: Intel MKL and CUDA

- Freely available and the build recipes are open source
  - Enterprise support is available (see right)

- Even provides a GUI for managing environments

**DATA SCIENCE LIBRARIES**

- **Data Science IDEs**
  - jupyter
  - spyder
  - jupyterlab
  - RStudio

- **Analytics & Scientific Computing**
  - NumPy
  - SciPy
  - Numba
  - pandas
  - DASK

- **Visualization**
  - Bokeh
  - HoloViews
  - Datashader
  - matplotlib

- **Machine Learning**
  - TensorFlow
  - Theano
  - H2O
  - sklearn

...and many more!
- Packages are installed from “channels”
- Anaconda Distribution
  - Anaconda.org
    - Provides 3GB of storage (soft limit, hard limit isn’t clear but 5-10GB)
    - “Contact us for additional storage”
- Custom repositories
  - Any HTTP(S) url can be used as a channel provided the correct structure is present
Community maintained channel  
- Based around a GitHub organisation (over 1,100 members)
- Various CI providers are used for builds
- Anaconda Inc. provides storage, bandwidth and general support

Contains over 6,000 packages  
- One Git repository per package
- Maintainers are listed in the recipe
- New packages are added by making a pull request

Lots of automation  
- General maintenance
- Monitor for new releases and make pull requests
- **Bioconda** [https://bioconda.github.io](https://bioconda.github.io)
  - Builds on top of conda-forge
  - Contains over 6,000 packages for bioinformatics research
  - Around 600 contributors and 450 GitHub organisation members
  - Similar to conda-forge (Anaconda Inc. provide hosting, use CI to build packages)
  - Publication about the initiative [doi:10.1038/s41592-018-0046-7](https://doi.org/10.1038/s41592-018-0046-7)

- **AstroConda** [https://astroconda.readthedocs.io](https://astroconda.readthedocs.io)
  - Maintained by Space Telescope Science Institute in Baltimore, Maryland
  - Only depends on the default Anaconda channel
  - Contains over 1,500 packages
  - Hosted independently of Anaconda Inc. - [http://ssb.stsci.edu/astroconda](http://ssb.stsci.edu/astroconda)

- **AstroPy seem to be setting up something similar**
  - [https://github.com/astropy/conda-channel-astropy](https://github.com/astropy/conda-channel-astropy)
Building packages
A package is built from a “recipe” which is a folder containing a file `meta.yaml`

- Jinja2 templates are used for fancy features

Defined with cross-compilation in mind

- Build, Host and Run sections of the dependencies
- Runtime dependencies can be propagated
- These can be explicitly ignored if needed

Builds are normally run a script: `build.[sh|bat]`

- Creates “build” and “install” conda environments
- Package is then tested by installing to a new environment

Various check made afterwards

- Overlinking, undefined dependencies, RPATH patching, ...

Metadata (conda-forge uses this to set the GitHub access rights)
Originally relied on compilers installed on the system*
  - Use CentOS 5 for builds and rely on ABI versioning
  - Eventually packaged gcc 4.8.5 because C++11 support was needed

Took a long time to figure out a proper solution
  - In October 2017 Anaconda 5.0 fixed this situation
  - conda-forge only finished their migration 2 weeks ago
  - Anaconda Distribution and conda-forge are harmonising

Now much more maintainable for future updates
  - First release included gcc 7.3.0
  - gcc 8.2.0 is already available

Based around cross-compiling

*This slide might get the details wrong
Since conda-build 3 the Anaconda distribution has used CDTs

- Repackage libraries from CentOS 6 and move them to the sysroot of conda’s pseudo-cross compilers
- Used at build time for linking
- Use libraries from the host at run time
- Rely on ABI symbol versioning

Why?

- Lack of resources to create a recipe for a robust binary
- Some libraries might have better performance when taken from the host

conda-forge prefers to use it’s own libraries

- I think they’re currently trying to harmonise
- OpenGL is a library where performance is a consideration
  - Currently waiting for someone to find enough time to test if it makes a difference
- Builds MUST be relocatable for conda

- Achieved by:
  - The usual RPATH tricks
  - Build and install to an environment with a very long path (255 characters)
  - When installing:
    - Text files: Effectively run sed on each file
    - Binary files: Effectively run sed on each file and append null bytes

- Mostly works, but occasionally needs some extra work
  - Example from patching ROOT’s fork of clang:
    - Returning a constant path as a std::string doesn’t work for different string lengths
  - Example from clang:

```
1 std::string Linux::computeSysRoot() const {  
2     return "/conda/conda-bld/clangdev-1234534/x86_64-conda_cos6-linux-gnu/sysroot";
3 }  
```

```cpp
extern "C"
{
    const char *conda_sysroot_directory();
```
➤ A huge number of people in HEP already use conda

➤ It has even been by some experiments
  ➤ XENON1T: https://indico.cern.ch/event/694818/contributions/2986393/
  ➤ NEXT: https://indico.cern.ch/event/694818/contributions/2985799/
  ➤ This was with the old compilers

➤ Lack of a ROOT package is inconvenient
  ➤ Netherlands eScience Center built a package a few years ago
    ➤ Stopped being maintained a while ago
    ➤ Never worked reliably outside of RHEL 6 based distributions

➤ There is already a cling package inside conda-forge
  ➤ Completely independent of HEP, used for the xeus-cling Jupyter kernel
The day before conda-forge moved to the new compilers ROOT was added!

Everything should work on any Linux distribution (macOS is a WIP)

GCC 7.3.0 included when installing (For now: Binaries are prefixed so $CC, $CXX, $LD... must be used)

ROOT 6.16.00 was added today for Python 2.7, 3.6 and 3.7
  ➤ Was delayed due to making some changes to the clang recipe
  ➤ Release day updates are reasonable in future

Developed by Chris Burr, Enrico Guiraud, Henry Schreiner with help from Axel Naumann, Danilo Piparo, Guilherme Amadio and probably others whom I apologise for forgetting
➤ conda-forge doesn’t support unreleased versions
  ➤ Already struggle for CI resources sometimes (Travis CI, Circle CI, Appveyor and soon Azure)

➤ Builds of the conda package will be added to ROOT’s nightlies soon

➤ Distributing the nightly builds of ROOT is potentially useful
  ➤ Will likely upload the binaries from Jenkins to a dedicated channel
  ➤ Anaconda.org channels cannot contain the word “root”
  ➤ Have set up a “HEP” channel will probably be used instead
Conda is very young
- Lots of low hanging fruit to work on
- Hacks that were used to get Anaconda working are being replaced with better solutions
  - Compiler infrastructure already replaced
  - Package file format: Version 2 proposal
- Anaconda Inc. is investing heavily in conda-forge and is very open to external contributions
- NUMFOCUS sponsors conda-forge

Ultimately it doesn’t matter...
- The wider community has already decided to use conda
- Later stages of HEP analysis are going to keep using it

Also supports: build variants, flexible version pinning, exporting/importing environments, …
Questions?
Conda builds everything* for macOS 10.9

The macOS SDK cannot be distributed (legally)

What should compilers distributed with conda use?
  - Especially now the SDK isn’t available at / (macOS 10.14+)

I’ve patched clang to
  - Look in common SDK locations
  - Set MACOSX_DEPLOYMENT_TARGET=10.9
  - See here and here for details

*A handful of recipes force a newer SDK when 10.9 isn’t supported anymore