Simplifying Scientific Python Package Installation and Usage

Amiya Maji
amaji@purdue.edu
Motivation

- How can I install tensorflow/pytorch?
- Why can’t I use tensorflow in Jupyter?
- Why is my Jupyter session not loading?
- Why is “sudo pip install” asking for password?
- Why can’t I import the libraries I installed?
- How can I share my environment with a colleague?
- Various conda related issues ...

My Python environment has become so degraded that my laptop has been declared a Superfund site.

Source: https://xkcd.com/1987/
Contributions

Managing and using virtual environments are challenging for novice users

Simplify and streamline installation of Python packages

• **Simplify** management of virtual environments
• **Collect** best practices and configurations from sites
• **Flexible** activation via modules

Empower interactive Python users
Contents

Challenges
Best practices
Need for automation
Motivation for conda-env-mod
Conda-env-mod
Success stories
Discussions
## Challenges

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Non-root installation isolated from base Python</td>
</tr>
<tr>
<td>2</td>
<td>Package documentation assuming root access</td>
</tr>
<tr>
<td>3</td>
<td>Complex dependencies, often on system libraries</td>
</tr>
<tr>
<td>4</td>
<td>Frequent package updates</td>
</tr>
<tr>
<td>5</td>
<td>Incompatible or missing dependencies</td>
</tr>
<tr>
<td>6</td>
<td>Updating packages later can break existing environment</td>
</tr>
<tr>
<td>7</td>
<td>Need to share installations with colleagues</td>
</tr>
<tr>
<td>8</td>
<td>Need to use custom packages in Jupyter notebooks</td>
</tr>
<tr>
<td>9</td>
<td>Filesystem and IO issues for parallel computing</td>
</tr>
<tr>
<td>10</td>
<td>Security policy compliance</td>
</tr>
</tbody>
</table>
Review of HPC Python docs

Public documentations
• National Labs
• University HPC centers
• International HPC centers

Common themes
• Use virtual environments (Conda, venv, virtualenv)
• Install from source
• Do not install in $HOME
• Python with MPI needs care
• Complex workflows need simplification
  • Install mpi4py
  • Create Jupyter Kernel

Dissimilarities
• Varying degrees of details
• Opinions about binary packages/Conda
# Common use-cases

- How do I install a Python package
- How do I use an installed Python package in my code
- How do I install and use a Python package in Jupyter notebooks
- How do I list the Python environments that I have created
- How do I delete a Python environment that I have created
- How do I list the packages that I have installed
- How do I update a package that I have installed
- How do I share my environment with a colleague
- How do I recreate an existing environment
Tasks for managing Python packages

Colored arrows indicate steps for a specific workflow.
<table>
<thead>
<tr>
<th>Need for simplification and automation</th>
</tr>
</thead>
</table>
| Many best-practices can be achieved with configuration files | • pip.conf  
• condarc |
| Some workflows require automation | • Create Jupyter Kernel  
• Install mpi4py, h5py etc. |
| Set sensible default values | • Default location of environments  
• Default package cache  
• Default threading  
• Do not use user site-packages |
Contents

- Challenges
- Best practices
- Need for automation
- Motivation for conda-env-mod
- Conda-env-mod
- Success stories
- Discussions
**conda-env-mod: Install Python packages**

- create myenv
- conda create
- create module
- create kernel
- module load
- pip install
- conda install
Demo
# Supported operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>Name, Prefix, YAML specifiction</td>
</tr>
<tr>
<td>export</td>
<td>YAML specification</td>
</tr>
<tr>
<td>module</td>
<td>Lmod, Tcl</td>
</tr>
<tr>
<td>kernel</td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
</tr>
<tr>
<td>delete</td>
<td></td>
</tr>
</tbody>
</table>
Benefits of environment modules

Environment modules are powerful tools
Programmatic way to set up runtime environment
Users can start with a clean environment and load modules on demand
Modules take care of setting appropriate variables (PATH, PYTHONPATH, etc.)

HPC center staff can incorporate best-practices into modules

Track dependencies on system modules

Detailed help messages

Other benefits
Avoid conda init and conda activate
No more polluting your bashrc
### Other benefits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
</table>
| Sharing Python installations within a cluster | One person manages the environment  
Others load a module/kernel to use it |
| Sites can customize module file/kernel templates |  |
| Allows stacked environments | Environment with stable packages (module X)  
Environment with experimental packages (module Y)  
Load modules X and Y  
Caveat: Need compatible Python version |
| Python package developers can easily share environments | environment.yaml  
modules files  
Jupyter kernel |
Contents

Challenges
Best practices
Need for automation
Motivation for conda-env-mod
Conda-env-mod
Success stories
Discussions
Success stories

• Deep Learning package installation
  • 11 applications (multiple versions)
  • 3 Python versions
  • CPU and GPU version
• Shared Python environment for teaching
  • Instructor uses conda-env-mod to install packages
  • Students load modules and Jupyter Kernels
  • 12+ course, 1500+ students (between 2019-20)
  • Data Science, Atmospheric Science, Molecular Chemistry, Library Science
• Shared Python environment for research groups
  • New feature requests
Conclusion

Broad impacts

• Capture best practices for scientific Python application installation
• Engage the scientific Python community for better packaging
• Improve scientific productivity, reduce user errors
• Help interactive Python use
  • JupyterHub
  • Gateway/Open OnDemand

Download

• https://github.com/amaji/conda-env-mod

Best practices document

• https://hpc-python-solutions.readthedocs.io/

Contributions are welcome!
Acknowledgements

This work is partially sponsored by the Better Scientific Software (BSSw) Fellowship sponsored by the DOE and the NSF.

Contributors

- Lev Gorenstein
- Zihan Xu

Big thanks to

- Dr. Hai Ah Nam (LBL)
- Lisa Frerichs (Krell)
Applications are open for the 2024 BSSw Fellowship Program

The **Better Scientific Software (BSSw) Fellowship Program** provides recognition and funding for leaders and advocates of high-quality scientific software who foster practices, processes, and tools to improve scientific software productivity and sustainability.

Each 2024 BSSw Fellow will receive up to $25,000 for an activity that promotes better scientific software. Activities can include organizing a workshop, preparing a tutorial, or creating content to engage the scientific software community, including broadening participation or promoting diversity, equity, and inclusion.

**Application deadline**: Applications for the 2024 BSSw Fellowship Program are being accepted through **September 29, 2023**

- Mailing List: https://bssw.io/pages/receive-our-email-digest
- More details: https://bssw.io/fellowship
Questions
Thank You

amaji@purdue.edu