When to use Conda and when to use pip

Conda and pip are both package managers for Python, but they have slightly different use cases and strengths. Here's a general guideline on when to use Conda and when to use pip:

Use Conda when:

1. **Managing Environments:** Conda is particularly strong when it comes to creating and managing isolated environments. Environments allow you to have different sets of packages and Python versions for different projects. This is especially useful when working on projects with conflicting package dependencies.
2. **Cross-Platform Compatibility:** Conda handles dependencies and installations in a way that often makes it easier to ensure cross-platform compatibility, especially for packages that rely on system-level libraries.
3. **Non-Python Dependencies:** If your project requires non-Python libraries or packages, Conda can handle these dependencies more effectively compared to pip.
4. **Complex Data Science Stack:** Conda is often preferred for data science and scientific computing projects due to its ability to manage both Python and non-Python dependencies commonly found in these stacks.

Use pip when:

1. **Package Availability:** pip has a larger and more comprehensive package repository, known as the Python Package Index (PyPI). If the package you need is available on PyPI, it's often easier and more straightforward to use pip.
2. **Python-Specific Packages:** For packages that are primarily focused on Python functionality and don't have complex non-Python dependencies, pip is a more natural choice.
3. **Virtual Environments:** While Conda also supports virtual environments, pip is the default package manager for creating virtual environments using the built-in **venv** module in Python.
4. **Integration with Build Systems:** If you're working on a project where the build and deployment processes rely heavily on pip, using Conda might introduce unnecessary complexity.

In many cases, you can use both Conda and pip together. For example, you might use Conda to manage your environment and install some packages, while using pip to install packages that are specifically available on PyPI.

Keep in mind that the lines between Conda and pip have blurred over time, with each tool adding features that partially overlap with the other. The choice between Conda and pip depends on your specific project requirements, the packages you need, and your familiarity with the tools.