Our approach: JetScape Workflow and Considerations

<u>**Considerations and implementation/design philosophy:**</u>

- **minimize dependency on external libraries** concerning the core framework code (all used external sources are "small" and are all part of the "core framework")
- **C++11 style** —> no new/delete **smart pointers** (shared, weak and unique) —> simplifies memory management (reduces chances of memory leaks)
- **Object Orientated (OO) Framework (C++ class inheritance)**
- **Tasked-based** implementation: *Init(), Exec(), Clear() and Finish()* Helpful for "further" parallelization (see JetScapeEnergyLossManager as a test case)
- **Strict data encapsulation** between modules (<u>only</u> "share" what is needed!)
- **Signal/Slot** mechanism to *ensure data encapsulation*. Also elegant solution concerning switching between energy loss modules and "communication" with hydro
- **Clear and easy interface** for further "end-user" developer, inherit from proper base class and overload the "JetScape interface functions" and "data structures" —> **No real knowledge** of the framework itself is needed and importantly is hidden —> **Safety!!!**