Steering and distribution for CORSIKA 8

Lukas Nellen
ICN-UNAM
lukas@nucleares.unam.mx

Steering

- CORSIKA 8 features implemented in library
  - Main needed for steering
- Can use with different main programs
- Flexibility for steering
  - Cards
  - Programmable main

Python for steering

- Implement main in python
  - Seems to be consensus

- Flexbile
- Easy to change main
- Special purpose mains
  - e.g. as a service
  - controllable via zero MQ

- Provide `Configuration` hierarchy
  - Fill C++ base tree of maps/vectors
  - Have C++ read python structure

Types of users

- Regular users
  - Want standard, stable configurations
- Feature developers
  - Need to be able to provide configurability for their features
- Core developers
  - Don’t want restrictions

Standard configuration(s)

- Don’t overwhelm users with choices
  - Cuts
  - Interaction parameters
  - …
- Named configurations
  - For use cases
  - For global choices
  - For collaborations
    - Infrastructure to share
    - Makes results more reproducable
    - Allows for more feedback from experts

Idea for simplistic Example

```
import corsika
from corsika import configuration
configuration.select('auger')

#alternative
configuration.gamma = -2
configuration.Emax = 10**20
configuration.Emin = 10**18
configuration.nShowers = 1
configuration.outDir = '.'
configuration.outFormat = 'lib_01_{}.dat'

# for modules
configuration.sibyll.xyx = …

corsika.run() # or loop over showers
```

Card driven running

- Needed?
- Configure in python script sufficient?

- Can be implemented in python
  - e.g., YAML based
  - JSON
  - INI format

More complicated use case

- Read (part of) configuration from steering DB
- Extract information
  - Shower information
  - Run summaries
  - Store in meta-data DB
- Glue to integrate with external steering
  - again: python is already standard choice
- Stack filling
  - Easy to integrate pre-interaction
  - Modified first interaction

Requirement: validation and debugging

- How do we debug configurations?
- Errors
  - Non-existing variables
  - Missing required variables
- Possible problems
  - Changes to defaults
- Validation
  - Should all happen at startup
  - Avoid delayed failure due to configuration errors

Global configuration

- need coconut-like script

- How much configuration needs re-compiling
  - Can we provide binary
  - Require user re-compilation

- For top level: produce different libraries
  - corsika_qgsjet
  - corsika_sibyll
  - choice at top-level config script

```
import corsika_qgsjet as corsika
from corsika import configuration
...
```

Distribution

- Source code
- Packages for distributions
  - Probably too much effort
- Static binaries / libraries
  - Not a good solution to interact with python

- Containers
  - With binaries
    - Feasibility depends on combinatorics of choices
  - For building, with script
  - singularity containers (?)
    - user installable
    - grid requirement

Flexibility vs speed

- Current design in CORSIKA 8: Compile-time polymorphism
  - Avoids virtual function overheads
  - All implementation have to be known at compile time
- Normal (virtual function) polymorphism
  - More flexibility
- Have enough physics in code
- Ready to try different ideas
  - Benchmarking needed
- Choice affects how we can distribute code

Caveat: docker

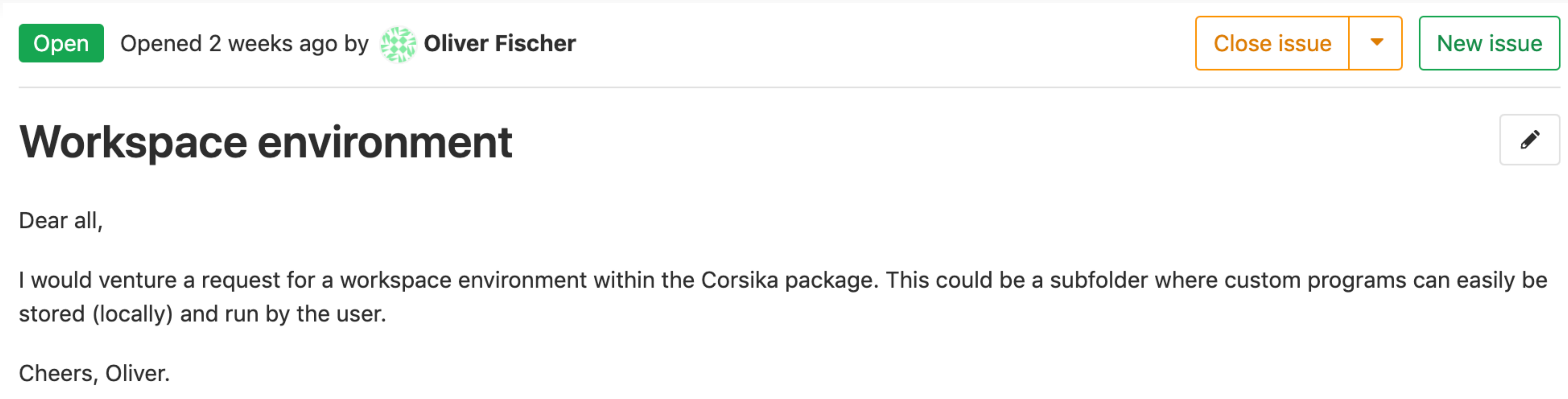
- Standard container solution
- Great for single user setup
- Used for testing
  - Easy to provide many platforms

- Security Problem: right to run containers implies root access
  - Could be (partially) solved by knowledgeable admin
  - Not suitable for shared systems

External packages

- Ship as `ThirdParty` code
  - Easy to ship
  - Might bloat source
- Required from distribution
  - Might require support from local admin
  - Difficult to control version
- Install tool
  - Not convenient for large, diverse user base
- In container
  - pre-install external packages
  - build-scripts available for developers: local environment

User workspace



Opened 2 weeks ago by Oliver Fischer

Workspace environment

Dear all,

I would venture a request for a workspace environment within the Corsika package. This could be a subfolder where custom programs can easily be stored (locally) and run by the user.

Cheers, Oliver.

- For user contributions
  - Fixed directory
  - Guidelines for users
- Document workflow
  - Users have to stay in sync with development
  - Merging after divergent development is painful
- Protection of new ideas

Privacy vs code sharing

- Would like to see code in repository
- How to protect contributor before publication
  - New ideas: might want to publish result
  - Use contribution as test bed and basis for publication
- Don’t want others to scoop contributors
- Partial protection of repo?
- Private repo?
  - Automated testing?
  - How to avoid problems with late merging?

Summary

- Consensus: python for programmable steering
- Would like standard configurations
- Distribution
  - Source, with build helper
  - container (singularity)
- Need support for contributors