Analyzing Projects : : **CHEAT SHEET**

Getting Started

Kaiāulu recommends a set of steps to analyze projects. These include the way to organize a project's folders and the way the analysis is configured for reproducibility.

Additional features facilitate project selection by different project demographics, reusing parts of Kaiāulu in other tools or server-side for parallelization.

Folder Organization

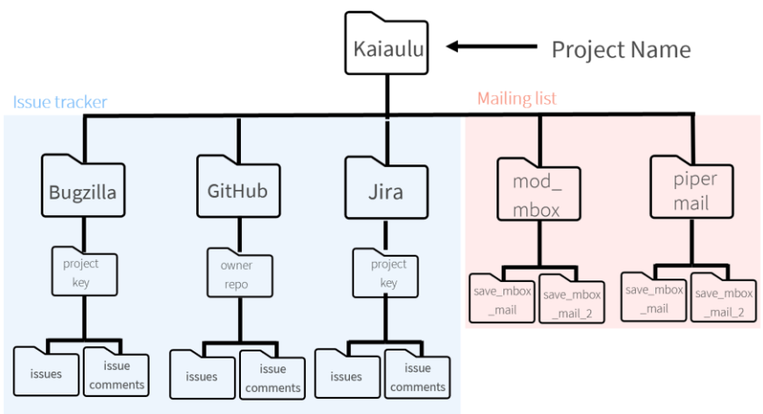
A project analysis is organized in sub-folders per data source provenance. A project folder can be initialized by the function:

create_file_directory()

The file organization is specified in the project configuration file. These, in turn, are used throughout the Notebooks via getter functions in Kaiāulu once the project configuration file is specified.

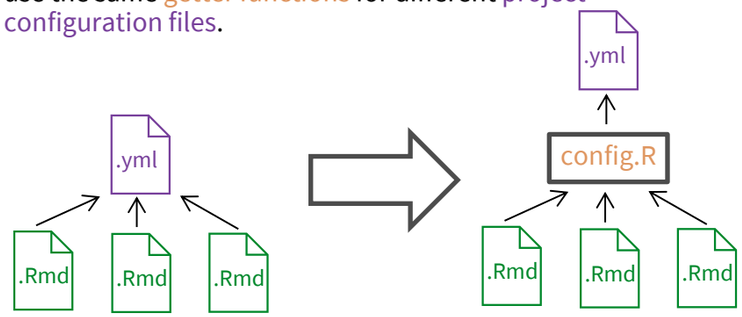
```
project_of_interest
├── jira
│   └── project_key
│       ├── issue_comments
│       └── issues
├── github
│   └── owner_repo
│       ├── discussion
│       ├── issue
│       ├── pr_comment
│       └── ...
├── bugzilla
│   └── project_key
│       ├── issue_comments
│       └── issues
├── mod_mbox
│   ├── save_mbox_mail
│   └── save_mbox_mail_2
└── pipermail
    ├── save_mbox_mail
    └── save_mbox_mail_2
```

- **mailing_list**
- **issue_tracker**



Accessing Configs from Notebooks

The getter functions in config.R centralize the process of gathering information from project configuration files (.yml) in Notebooks (.Rmd). This allows for different Notebooks to use the same getter functions for different project configuration files.



Without the getter functions from config.R, the Notebooks would require direct variable assignments to the project config information.

With the getter functions from config.R, the Notebooks can use the getter functions to acquire the project config information.

If the project config specification evolve in the future, only the corresponding getter function implementation needs to be updated, instead of all the dependent Notebooks and Exec scripts.

Selecting Projects

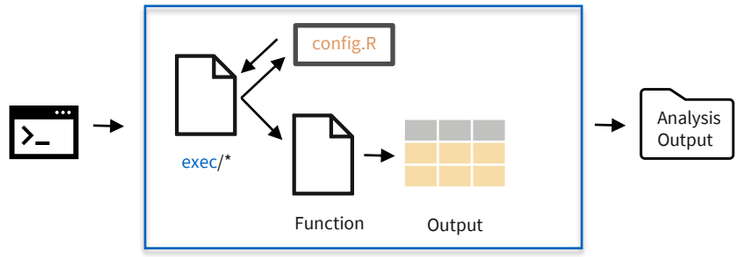
Project selection often involves tedious work in browsing different websites and repositories to identify project demographics, language, issue traceability, or bug labeling.

openhub_project_search.Rmd is a Notebook that showcases how Kaiāulu interfaces with OpenHub API to facilitate selecting open-source projects for studies. This Notebook demonstrates how to use the Kaiāulu OpenHub API interface to search for projects that meet specific criteria, streamlining the process of discovering open-source projects through OpenHub's database.

Once projects for analysis are decided upon, they can be documented as project configuration files, in turn accessed in Notebooks by getter functions.

Command Line Interface

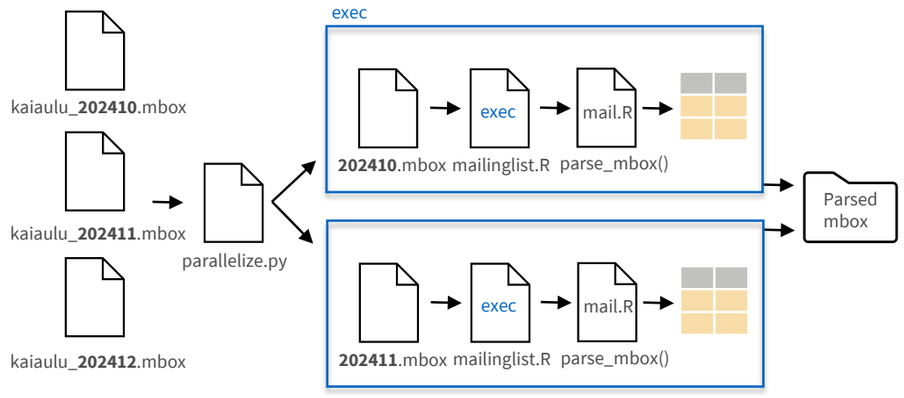
Exec scripts serve as interface between Kaiāulu functions and external tools, allowing a subset of capabilities to be accessed via the command line in other languages.



Executable files also utilize getter functions to access project configuration files, and Kaiāulu's API to externalize functionality. Exec scripts expect a combination of both project configuration files as input and optional arguments.

Parallelization

Through Exec scripts, Kaiāulu functions can be called in parallel, enabling concurrent analysis of large projects.



The parallelize.py script can be used with other Kaiāulu Exec scripts depending on the analysis needs.