Modular Parallelization: : CHEAT SHEET

About

The modular parallelization tool provides a set of functions to access project configurations using getter functions and search for open-source projects.

It also enables the parallelization of tasks through scripts, improving efficiency when managing and analyzing large projects.

The tool streamlines project management by organizing information at the project level through a consistent folder structure for project configuration files.

Folder Organization

Taking into consideration the ease of use for users, the project folder organization has been standardized under project keys.

A project folder can be initialized by the function:

create_file_directory()

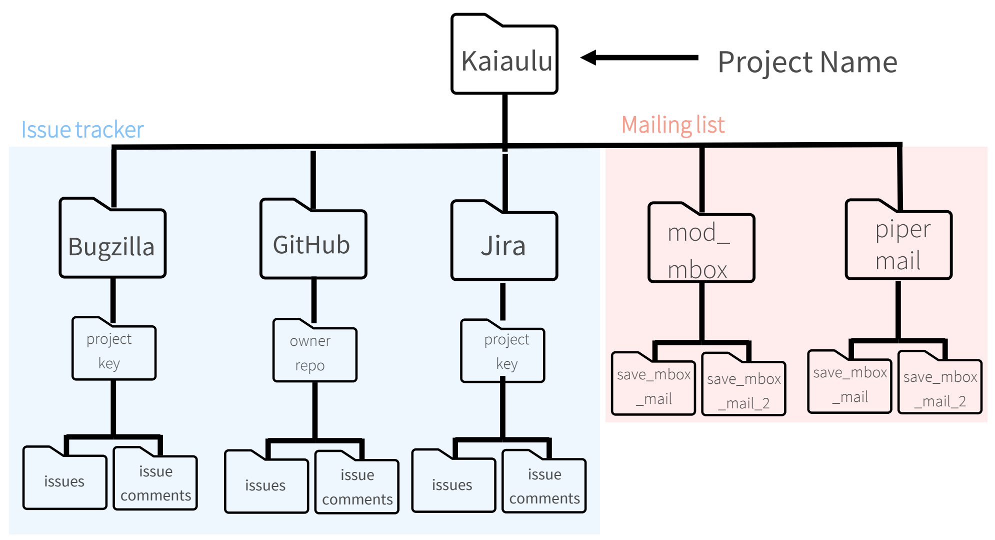
located in config.R.

The file organization is specified in the project configuration file

A user can specify their own file directory by changing this file.

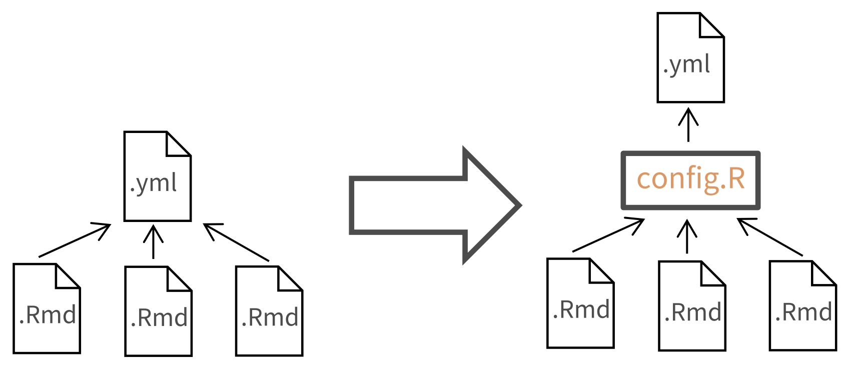
▪	mailing_list
▪	issue_tracker

```
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
```



Accessing Project Configs

The getter functions in config.R centralize the process of gathering information from configuration (.yml) files. This allows for different notebooks (.Rmd) 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 changes, only the corresponding getter function implementation needs to be updated, instead of all the dependent notebooks.

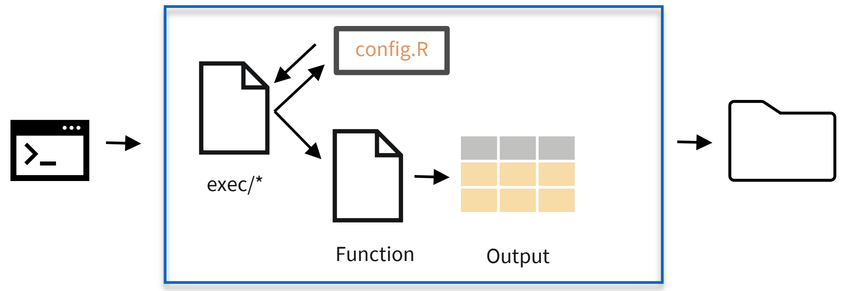
Selecting Projects

openhub_project_search.Rmd is a notebook that interfaces with OpenHub API to facilitate locating open-source projects for studies.

This notebook demonstrates how to use the Kaiāulu OpenHub API interface functions in config.R to search for projects that meet specific criteria, streamlining the process of discovering open-source projects through OpenHub's database.

Command Line Interface

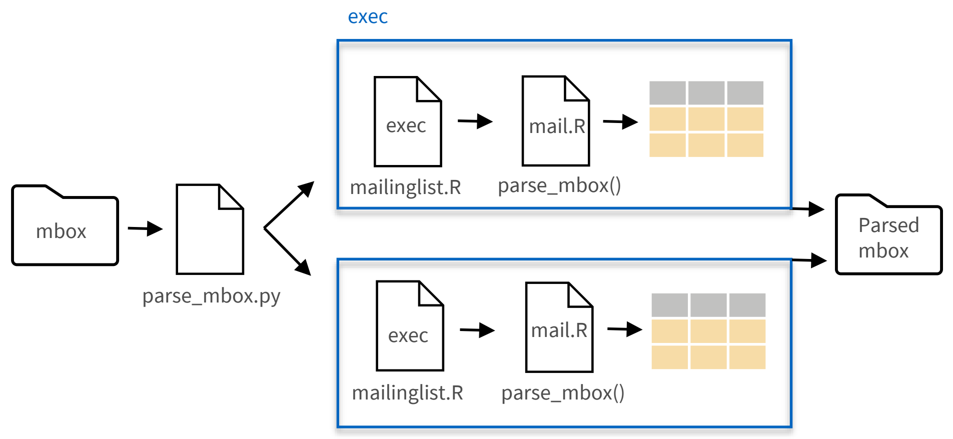
Exec scripts serve as interfaces between Kaiāulu functions and external tools, allowing capabilities to be accessed via the command line.



These scripts are highly reusable and adaptable to project-specific needs through configuration files that provide customizability. Exec scripts enable Kaiāulu to remain flexible, maintainable, and extensible.

Parallelization

Through exec scripts, Kaiāulu functions can be called in parallel, enabling the efficient analysis of large projects.



A Python script handles the parallelization, using ThreadPoolExecutor to parse multiple files concurrently. A new job is created for each file, and each job calls the exec script to use Kaiāulu's parsing function.

This method can be expanded to other tasks, by changing the exec file used, and handling the parsing of data according to the task's expected output format.

Kaiāulu