# Code Publishing Cheat Sheet

## Project

### General principles

- **Avoid code smells**: Remove any function or class that makes the code unreadable or difficult to maintain or extend.
- **Refactor Early and Often**: Continuously improve the code to make it simpler and more maintainable.
- **No dead code**: Remove any code that is not used or is not needed.
- **Better no comment than a faulty comment**: Avoid adding comments to explain code that is already clear.
- **Avoid the obvious comment**: Use comments to explain why you chose a particular approach, not what you did.

### Timeboxing

- **Scale-out via the process mode (separate process by container)**: Use containers to scale out your application.
- **Explicitly declare and isolate dependencies**: Make sure that your dependencies are clearly defined and isolated.
- **Strictly separate build and run stages**: Keep your build and run stages separate to avoid mixing concerns.

### Separation of concerns

- **DRY/AHA**: Do not repeat yourself. Abstract common functionality into reusable components.
- **Statelessness**: Ensure that your code is stateless to make it easier to test and maintain.

### Code comments

- **Avoid code comments**: Use comments sparingly to explain why a particular approach was chosen, not what was done.
- **Refactor Early and Often**: Continuously improve your code comments to make them more relevant and clear.
- **Limit lines per files**: Keep your code comments short and to the point.

### Packaging

- **An abstraction layer such as a Docker image can remove the hassle of dependencies (see Distribution below)**

### Git workflow

- **Wrap the body at 72 characters**: Make sure your commit messages are short and to the point.
- **Use the imperative mood in the subject line**: Use active voice to describe what you did.
- **Capitalize the subject line**: Make your commit messages stand out.
- **Limit the subject line to 50 characters**: Keep your commit messages concise.

### Tools

- **gitmoji**: Use git emojis to make your commit messages more fun and engaging.
- **Conventional-commit**: Use this tool to ensure that your commit messages follow the conventional commit message format.

### License

- **Chose a license**: Choose a license that is suitable for your project and make sure that it is clearly documented.

## Code

### Code documentation

- **Use may use standard version/development/changelog to changing and documentation related to releases**

### Code publishing

- **Use versioning**: Use versioning to track changes to your codebase.
- **Use documentation**: Use documentation to explain how to use your code.

### Testing

- **Testing key at each release of the software, to avoid breaking the code**: Use automated testing to ensure that your code is stable.

### Documentation

- **WHY? Instead of running the tests, code analysis and/or generating the documentation manually, this can be done automatically by a continuous integration pipeline**

## Dev Tools

### Continuous Integration / Continuous Delivery

- **WHY? Instead of running the tests, code analysis and/or generating the documentation manually, this can be done automatically by a continuous integration pipeline**

### Packaging

- **An abstraction layer such as a Docker image can remove the hassle of dependencies**: Use a Docker image to manage dependencies.

## Publish

### User Support

- **WHY? User support is very important to building a community of users around your software**

### License

- **WHY? Licenses are a legal contract between authors and users of a creative work, the authors can grant different levels of permission and liability under specified conditions**

### Resources

- **Funding**: You can find below potential interesting funding opportunities at ENAC, EPSI, Swiss or International level, selected by the Dean’s Office due to their connection with ENAC’s sustainability challenges.

- **Why Open Software (Open Science)**?: Don’t forget license file!

- **License**: Licenses are a legal contract between authors and users of a creative work, the authors can grant different levels of permission and liability under specified conditions.

- **User Support**: User support is very important to building a community of users around your software.

- **License**: Licenses are a legal contract between authors and users of a creative work, the authors can grant different levels of permission and liability under specified conditions.

- **Publications**: Open software journals such as open science.

- **Why Open Software (Open Science)**?: Don’t forget license file!

- **License**: Licenses are a legal contract between authors and users of a creative work, the authors can grant different levels of permission and liability under specified conditions.

- **User Support**: User support is very important to building a community of users around your software.