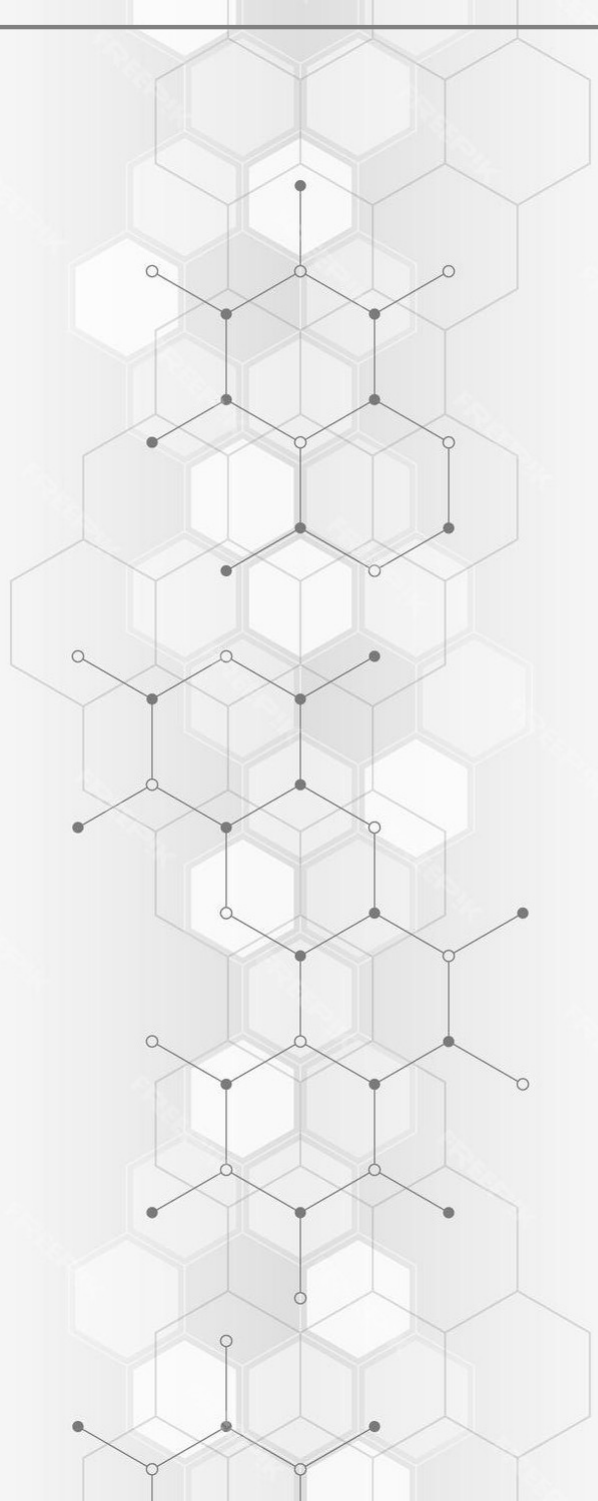
Recommended Programming Practices Practical Guide

Team Meeting - Network Dynamics and Computations

December 9, 2024



- Objectives
- Overview: Structure, Tools, Steps
- Initializing a Project

Objectives 3

Project - Definition

Cohesive set of code, resources, and related files organized to achieve a specific *goal*.

Not only code
Includes documentation, configuration files, tests...

Criteria for project quality?

Quality Criteria

Ultimate Goals

Reliability

Robustness, Reproducibility

Proximal Goals

Quality Criteria

Ultimate Goals

Reliability

Robustness, Reproducibility

Maintainability

Modifiability, Scalability

Proximal Goals

Quality Criteria

Ultimate Goals

Reliability

Robustness, Reproducibility

Maintainability

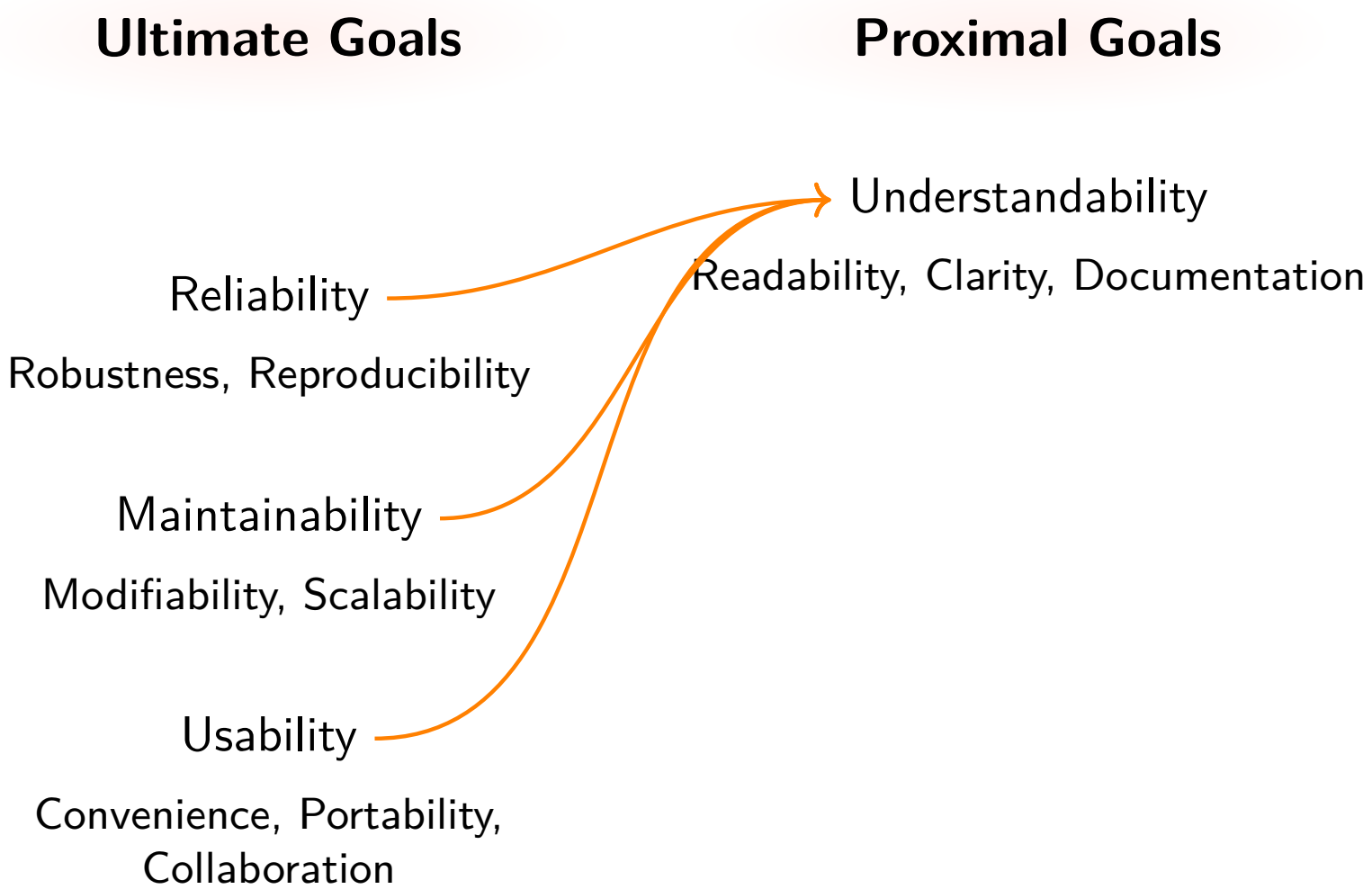
Modifiability, Scalability

Usability

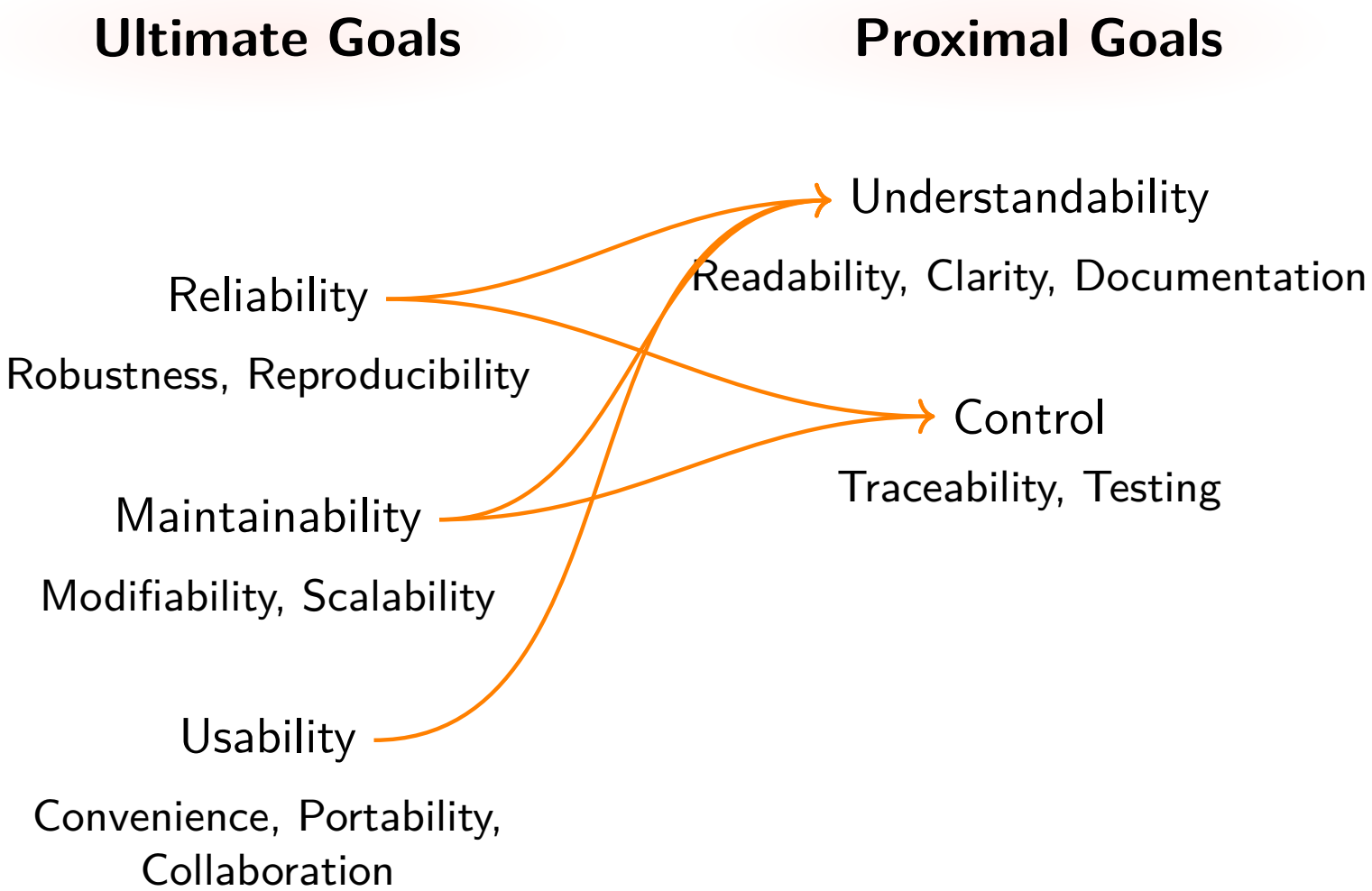
Convenience, Portability, Collaboration

Proximal Goals

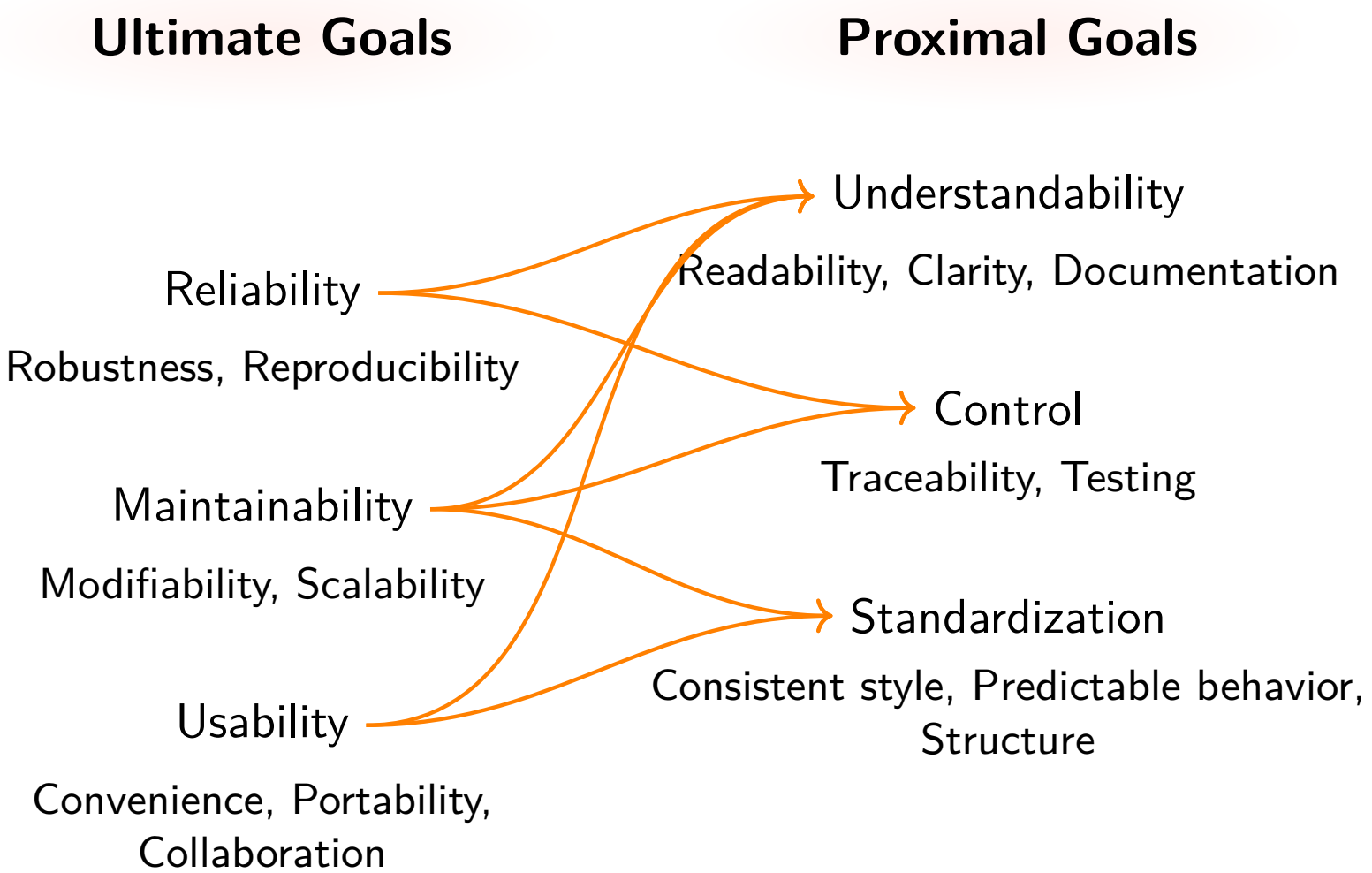
Quality Criteria



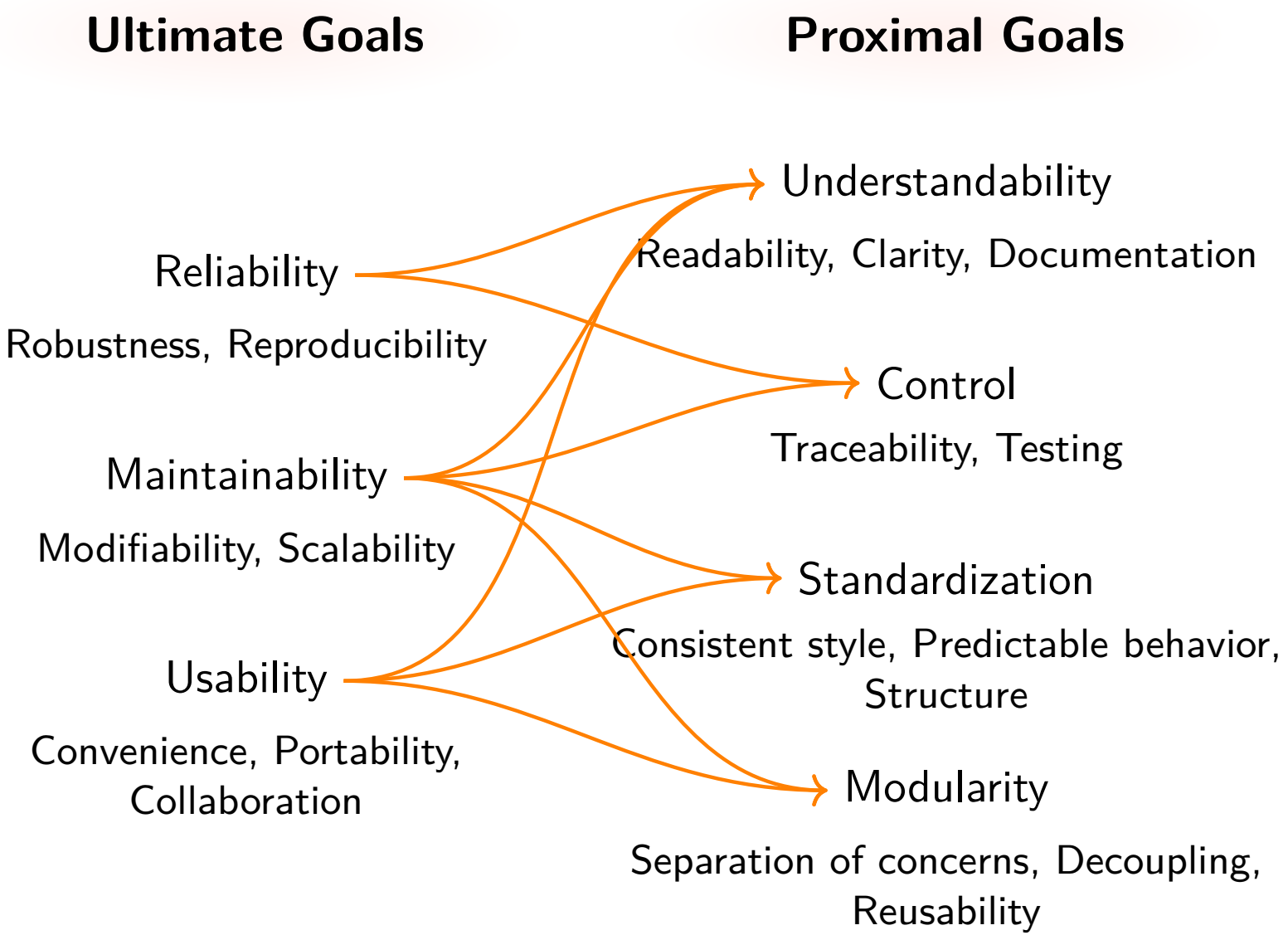
Quality Criteria



Quality Criteria



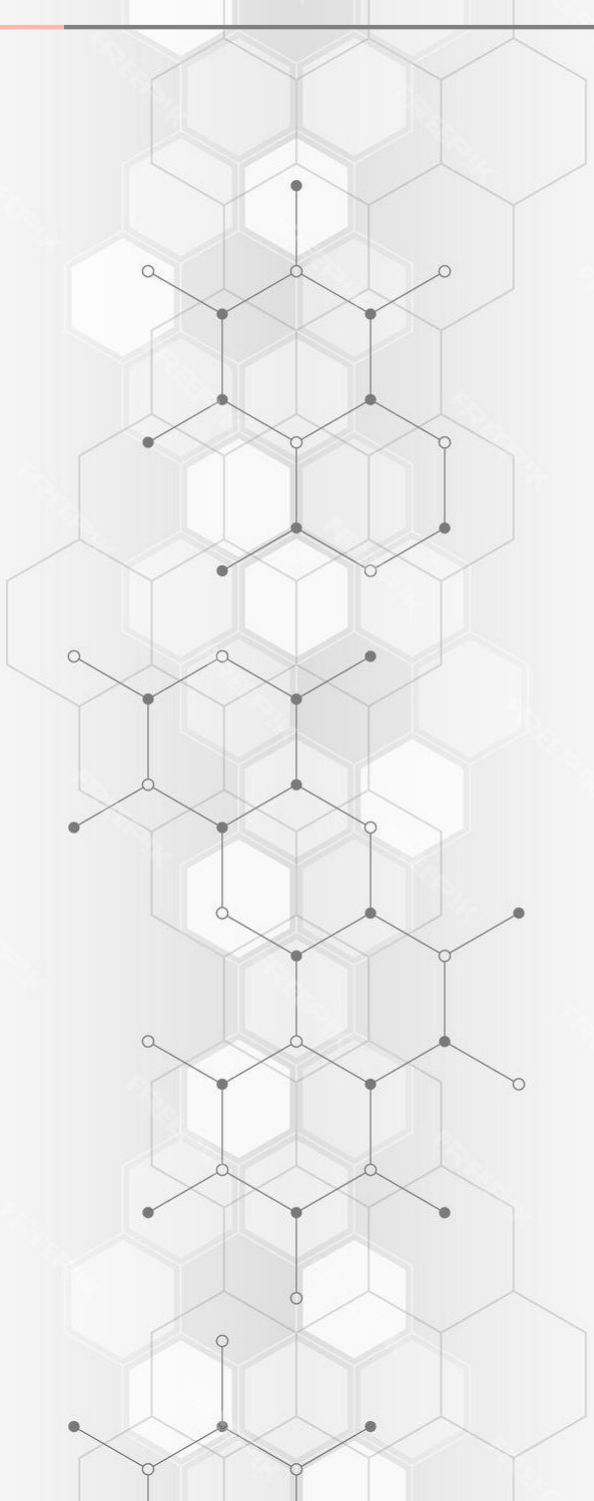
Quality Criteria



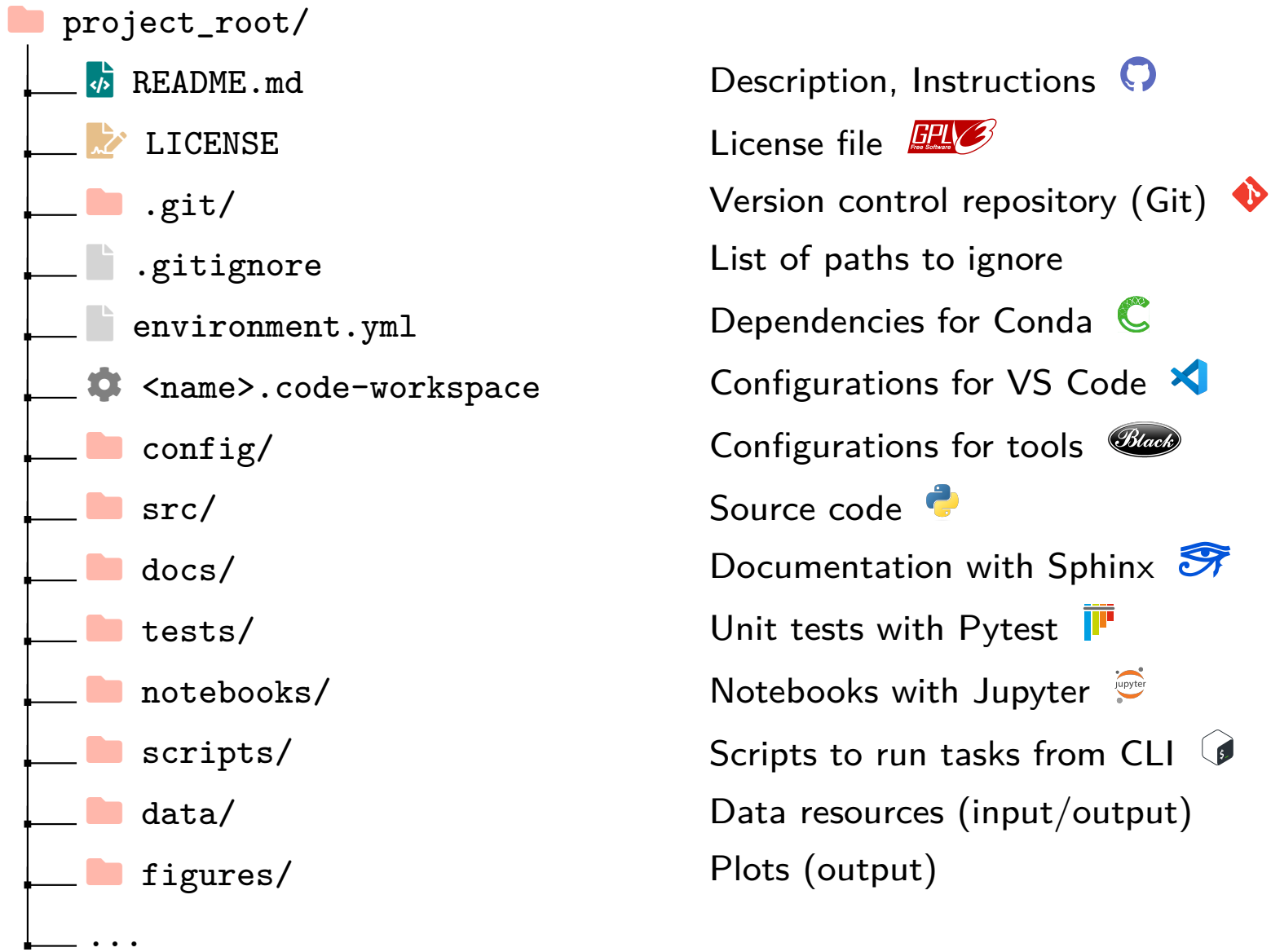
Collaboration

With others

With oneself in the future

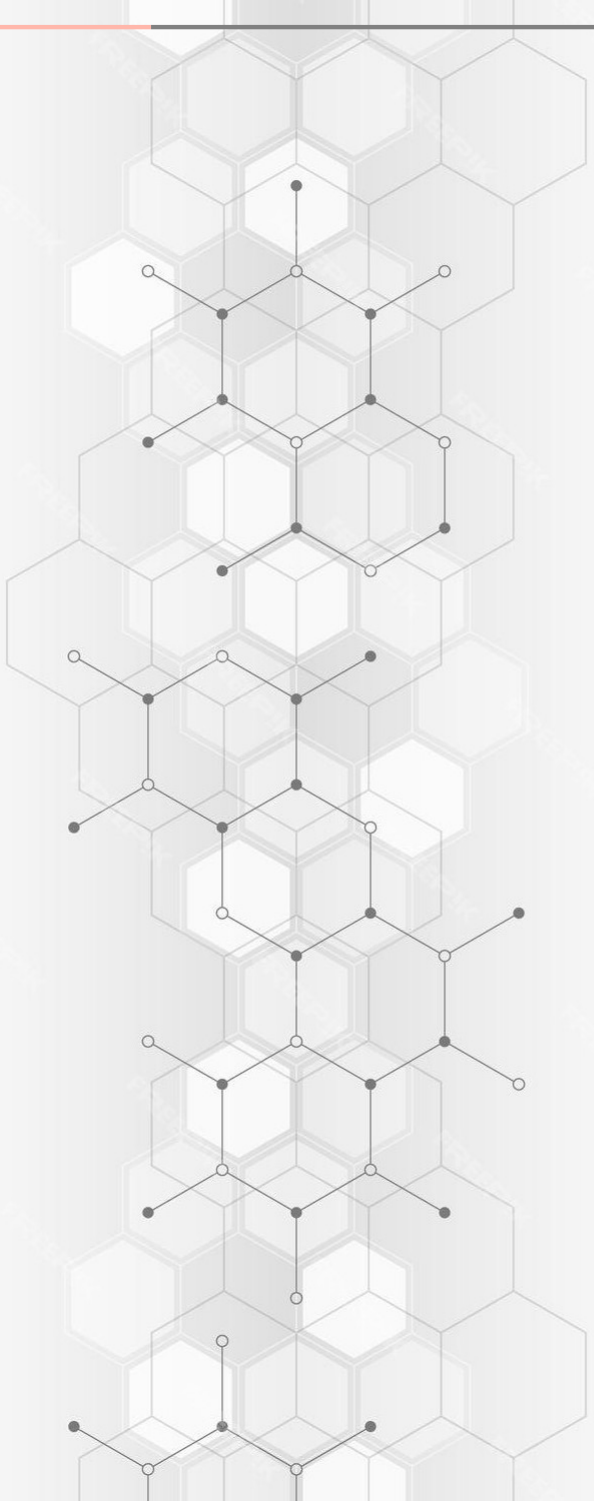


- Objectives
- Overview: Structure, Tools, Steps
- Initializing a Project


project_root/
README.md
Description, Instructions
LICENSE
License file
.git/
Version control repository (Git)
.gitignore
List of paths to ignore
environment.yml
Dependencies for Conda
<name>.code-workspace
Configurations for VS Code
config/
Configurations for tools
src/
Source code
docs/
Documentation with Sphinx
tests/
Unit tests with Pytest
notebooks/
Notebooks with Jupyter
scripts/
Scripts to run tasks from CLI
data/
Data resources (input/output)
figures/
Plots (output)
...

Project Life Cycle

Initialization	Development	Deployment
Repository	Implementation	Distribution
Environment	Testing	Reproduction
Workspace	Documentation	Maintenance
Tools		
Structure		



- Objectives
- Overview: Structure, Tools, Steps
- Initializing a Project

Initialization Workflow

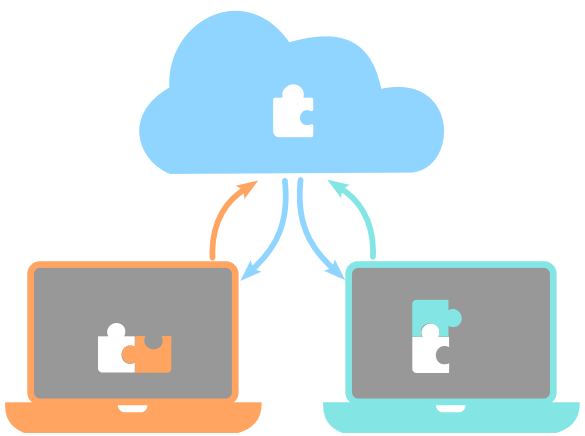
Initialization **Development** **Deployment**

- **Create Remote and Local Repositories**
- Establish Rules for the Project
- Construct the Project Structure
- Create a Virtual Environment
- Create a Code Workspace

Initialization Workflow

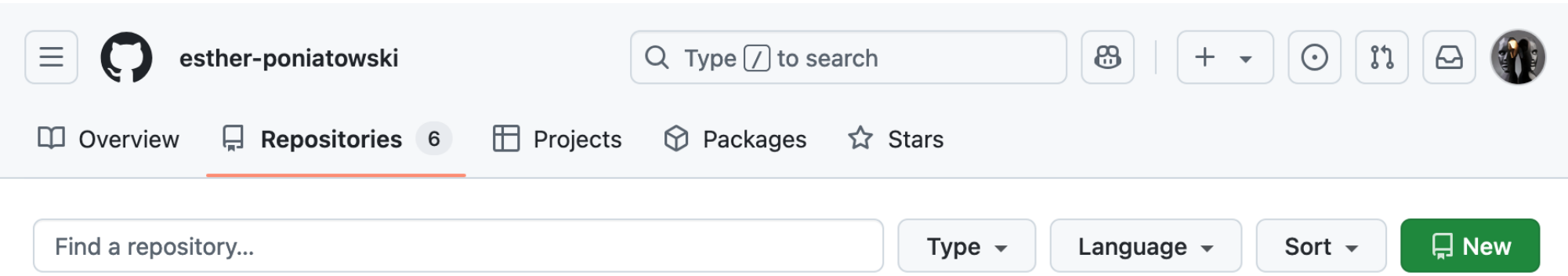
Create Remote and Local Repositories

Goal	Version-control, Tracking changes, Backup
Importance	Reliability, Traceability, Standardization
Tools	Git GitHub



Create Remote and Local Repositories

Create the Remote repository on GitHub (`origin`)



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * **Repository name ***

esther-poniatowski / ProjectName

ProjectName is available.

Great repository names are short and memorable. Need inspiration? How about **literate-train** ?

Description (optional)

- Public
  Anyone on the internet can see this repository. You choose who can commit.
- Private
  You choose who can see and commit to this repository.

Initialize this repository with:

- [x] Add a README file
  This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: GNU General Public License v3.0

A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

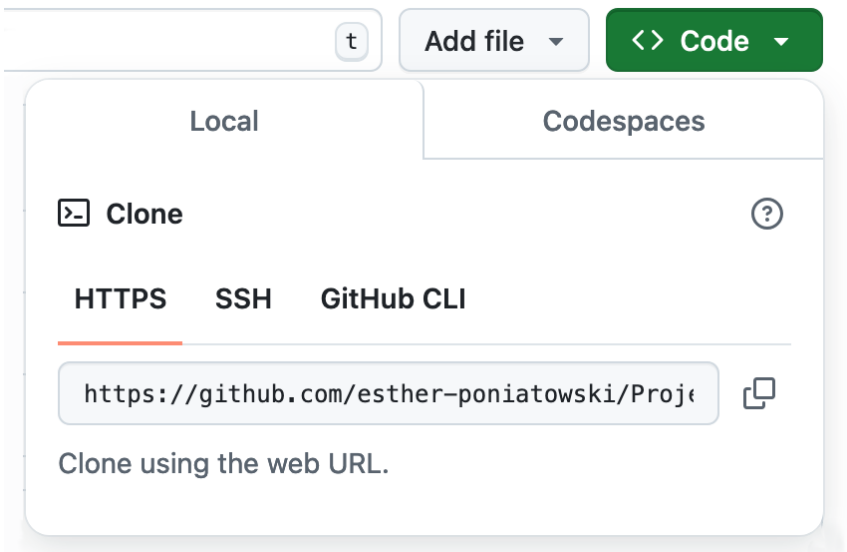
Create repository

Create Remote and Local Repositories

Clone the repository locally

Navigate to the location where the project should reside
(e.g. `$HOME/Projects`)

```
$ git clone 'https://github.com/user/repo.git'
```



Outcomes

- Copy of the remote repository in the current working directory.
- Name: Similar to the name of the cloned repository (by default).
- Content: All the repository files and history (`.git/`).

Remote repository associated with the local repository: `origin`

```
$ git remote -v
```

Notes

- The `.git/` folder is not visible in GitHub.
- The local repository inherits the full commit history of the remote repository, but from now both repositories can diverge (independent branches).

Create Remote and Local Repositories

Configure the local Git repository

Local Settings	Global Settings
Repository-specific	Apply to all repositories, if not overridden locally
Stored in `.git/config` (project folder)	Stored in `/.gitconfig`
Commands with `--local` flag or none	Commands with `--global` flag

Create Remote and Local Repositories

Configure the Local Repository

Check user profile (used to attribute commits):

```
$ git config user.name
$ git config user.email
```

Update if necessary:

```
$ git config user.name "Example Name"
$ git config user.email "exampleemail@domain.com"
```

Create Remote and Local Repositories

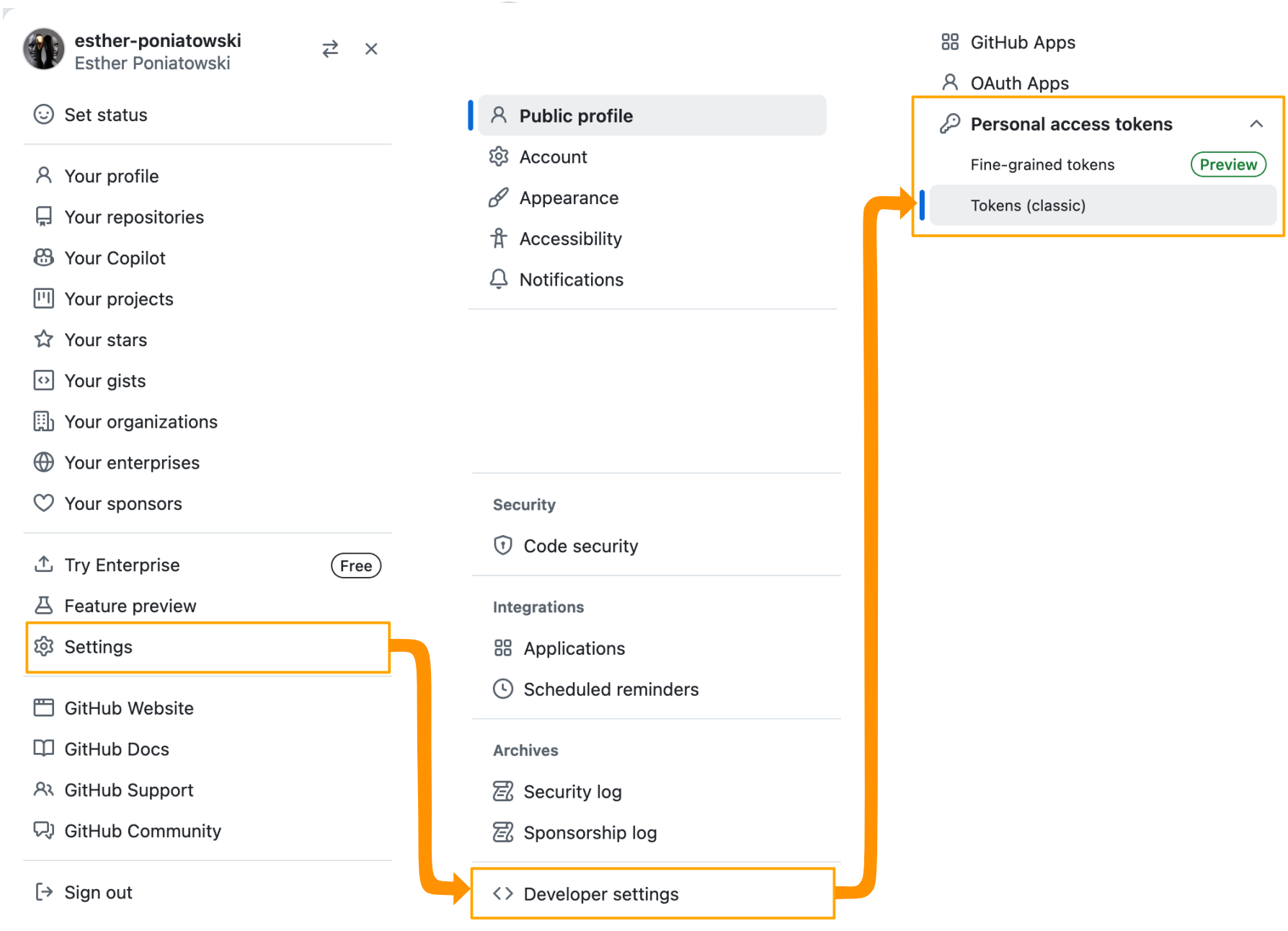
Set the Credentials

Pushing local changes to the remote repository requires authentication:

```
$ git push
Username for 'https://github.com': ...
```

Warning

GitHub removed support for password authentication in August 2021.
Alternative approach (among others): Personal access token.


esther-poniatowski
Esther Poniatowski
Set status
Your profile
Your repositories
Your Copilot
Your projects
Your stars
Your gists
Your organizations
Your enterprises
Your sponsors
Try Enterprise
Free
Feature preview
Settings
GitHub Website
GitHub Docs
GitHub Support
GitHub Community
Sign out
Public profile
Account
Appearance
Accessibility
Notifications
Security
Code security
Integrations
Applications
Scheduled reminders
Archives
Security log
Sponsorship log
Developer settings
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Preview
Tokens (classic)

New fine-grained personal access token Preview

Create a fine-grained, repository-scoped token suitable for personal API use and for using Git over HTTPS.

Token name *

ProjectNameToken

A unique name for this token. May be visible to resource owners or users with possession of the token.

Resource owner

esther-poniatowski

Expiration *

30 days The token will expire on Tue, Jan 7 2025

- 7 days
- ✓ 30 days
- 60 days
- 90 days
- Custom...
- No expiration

What is this token for?

Repository access

- ○ **Public Repositories (read-only)**
- ○ **All repositories**
  This applies to all current *and* future repositories you own.
  Also includes public repositories (read-only).
- ◉ **Only select repositories**
  Select at least one repository. Max 50 repositories.
  Also includes public repositories (read-only).

Select repositories

ProjectTemplate

esther-poniatowski/**ProjectTemplate**
no description

Permissions

Read our permissions documentation for information about specific permissions.

Repository permissions

Repository permissions permit access to repositories and related resources.

Actions
Workflows, workflow runs and artifacts.
Access: No access

Administration
Repository creation, deletion, settings, teams, and collaborators.
Access: No access

Attestations
Create and retrieve attestations for a repository.
Access: No access

Code scanning alerts
View and manage code scanning alerts.
Access: No access

Codespaces
Create, edit, delete and list Codespaces.
Access: No access

Codespaces lifecycle admin
Manage the lifecycle of Codespaces, including starting and stopping.
Access: No access

Codespaces metadata
Access Codespaces metadata including the devcontainers and machine type.
Access: No access

Codespaces secrets
Restrict Codespaces user secrets modifications to specific repositories.
Access: No access

Contents
Repository contents, commits, branches, downloads, releases, and merges.
Access: Read and write

Select an access level
- No access
- Read-only
- ✓ Read and write

Custom properties
View and set values for a repository's custom properties, when allowed by th

Dependabot alerts
Retrieve Dependabot alerts.

Create Remote and Local Repositories

Set the Credentials

.git-credentials

Tool	GitHub
Type	Plain Text
Location	./ (project root)
Content	Credentials for the remote repository.

Add credentials to the file:

```
1 https://<username>:<token>@github.com
```

Examples

```
1 https://esther-poniatowski:PERSONAL_ACCESS_TOKEN@github.com
```

Create Remote and Local Repositories

Set the Credentials

Use the credentials stored in the credential file:

```
$ git config credential.helper 'store --file=.git-credentials'
```

Outcomes

- For authentication, Git automatically uses the repository-specific credentials (user name and token) stored in the `.git-credentials` file.
- The user is not required to enter the personal access token manually.

Initialization Workflow

Initialization **Development** **Deployment**

- Create Remote and Local Repositories
- **Establish Rules for the Project**
- Construct the Project Structure
- Create a Virtual Environment
- Create a Code Workspace

Create a Repository

Give Project information and Instructions

README.md

Tool	GitHub
Type	Markdown
Location	./ (project root)
Content	Project description Installation and setup instructions Usage Contribution guidelines Contact information...

Create a Repository

Specify the Ignored Patterns

.gitignore

Tool	Git
Type	Plain Text
Location	`./` (project root)
Content	List of files and directory patterns to exclude from version control: private information (credentials), data, temporary files, build artifacts...

Create a Repository

Establish a Commit Message Template

.gitmessage

Tool	Git
Type	Plain Text
Location	./ (project root)
Content	Template for commit messages. Standardized format across contributors.

Create a Repository

Establish a Commit Message Template

General Structure

```
prefix(scope): Title (< 50 characters) ---------

Description: What and Why (not How) (< 72 characters per line)

Ref-Keyword: #ID

Signed-off-by: User Name <user.name@example.com>
```

Proposed Template

```
_():

Signed-off-by: User Name <user.name@example.com>
```

Prefixes for Commit Types

Conventional keywords to categorize the changes introduced by a commit. They can be used as filters to search the commit history.

`fix`	Correct a bug.
`feat`	Introduce a new feature or functionality.
`refact`	Refactor the code without changing its external behavior.
`test`	Add or update tests.
`docs`	Add or update documentation.
`style`	Improve formatting.
`perf`	Enhance performance.
`build`	Affect build components.
`chore`	Miscellaneous: manage dependencies, configure tools...
`revert`	Reverts a previous commit.
`merge`	Merge a branch.

Ref Keywords

Keywords recognized by the issue tracker on GitHub, to link commits to issues and to trigger actions (if applicable).

Keywords for Closing Issues

Automatically close an issue upon merging into the default branch.

`close`, `closes`, `closed` `fix`, `fixes`, `fixed`
`resolve`, `resolves`, `resolved`

Keywords for Referencing Issues

Provides context about the task addressed by the commit.

`references`, `refs` `related to` `see`
`duplicate of`

Examples

```
fix(compute_sum): Correct calculation of `chechsum`

Handle edge case where `a=0`.

Fixes: #123

Signed-off-by: Esther Poniatowski <eponiatowski@ens.psl.eu>
```

```
docs(README): Correct typos

- Indentation in lists in "Notes" section
- Name change: `batchLogbatchLog` -> `batchLog`

Signed-off-by: Esther Poniatowski <eponiatowski@ens.psl.eu>
```

```
feat(parser): Add support for .m files (Matlab)

Adapt the pipeline from the entry point to the `parse` function.

Signed-off-by: Esther Poniatowski <eponiatowski@ens.psl.eu>
```

Create a Repository

Establish a Commit Message Template

Retrieve the template from the remote repository and set it up locally:

```
$ git pull origin main
$ git config --local commit.template .gitmessage
```

Outcomes

- When committing, an editor opens and displays the template to edit (when running `git commit` without the `-m` option).

Notes

- The template is not used when committing in the remote repository.

Create a Repository

Establish an Issue Template

Issue Form

Tool	GitHub
Type	YAML
Location	`.github/ISSUE_TEMPLATE/general.yml` (on the remote repository)
Content	Template for issue submission Standardized format across contributors Align with the Commit Message Template

Create a Repository

Establish an Issue Template

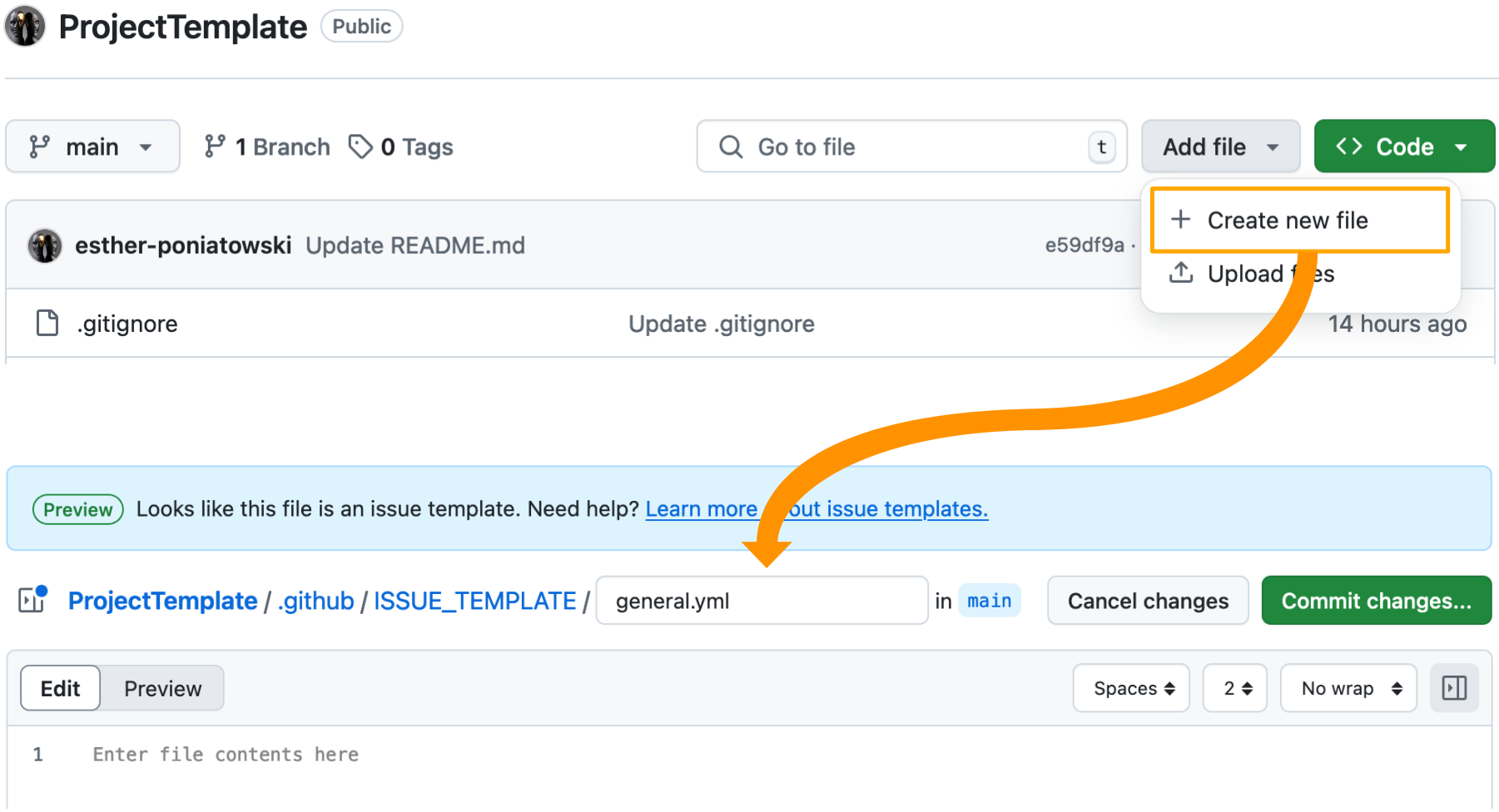
Recommendation

Use Issues for Self-Organization

- Structured to-do list for project management
- Prioritize tasks

Create a Repository

Establish an Issue Template



Create a Repository

Establish an Issue Template

General Structure

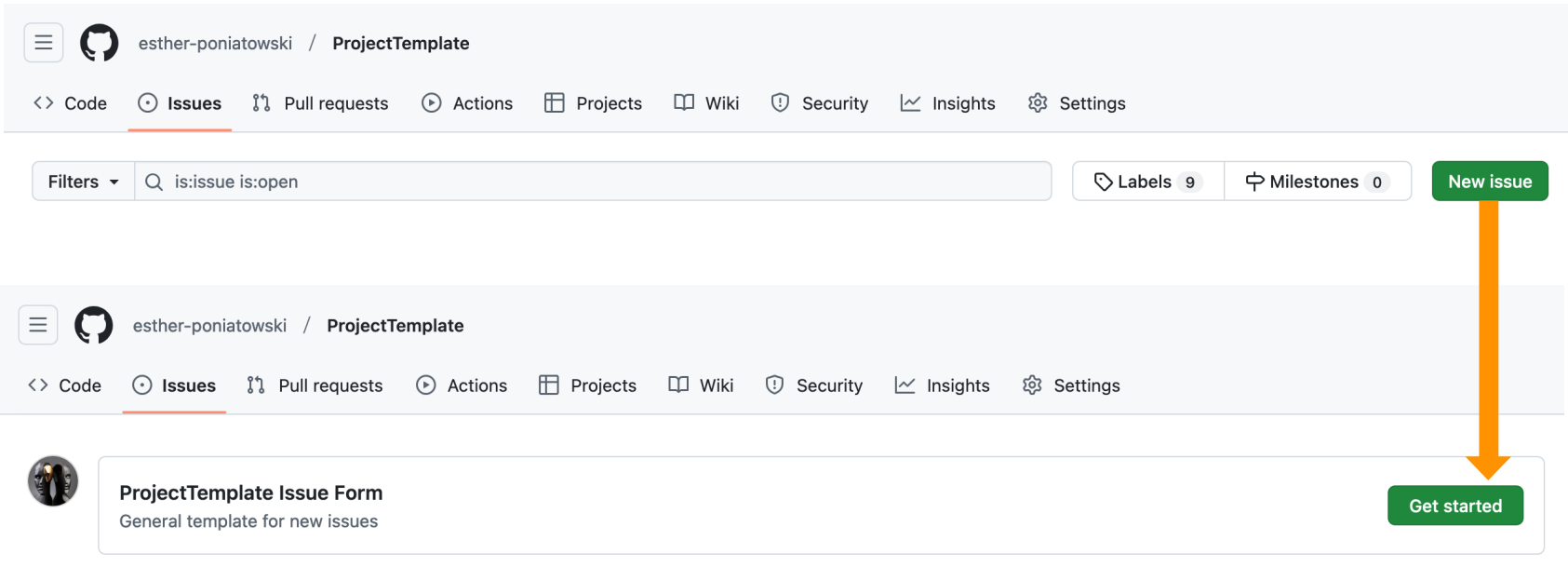
```
name: <name>
description: <description>
title: "_(): "
labels: ["feat"]
assignees: assignee-name
body:
    - type: <input | textarea | markdown | dropdown | chekboxes
        | poption>
    id: <identifier>
    attributes:
        <...>
    validations:
        required: false
```

Outcomes

ProjectTemplate / .github / ISSUE_TEMPLATE / **general.yml**

Preview This file is used as an Issue Form template. Give Feedback.

Select the issue form in the "New Issue" tab



Issue: ProjectTemplate Issue Form

General template for new issues. If this doesn't look right, choose a different type.

Add a title

Title

Scope

Related branch, files, code lines...

- [](url)

Description

Why, What

Acceptance Criteria

Requirements to close the issue

- [ ]

References

Related issues or PRs

-

Fields marked with an asterisk (*) are required.

Submit new issue

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Assignees

esther-poniatowski

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Shows branches and pull requests linked to this issue.

Helpful resources

GitHub Community Guidelines

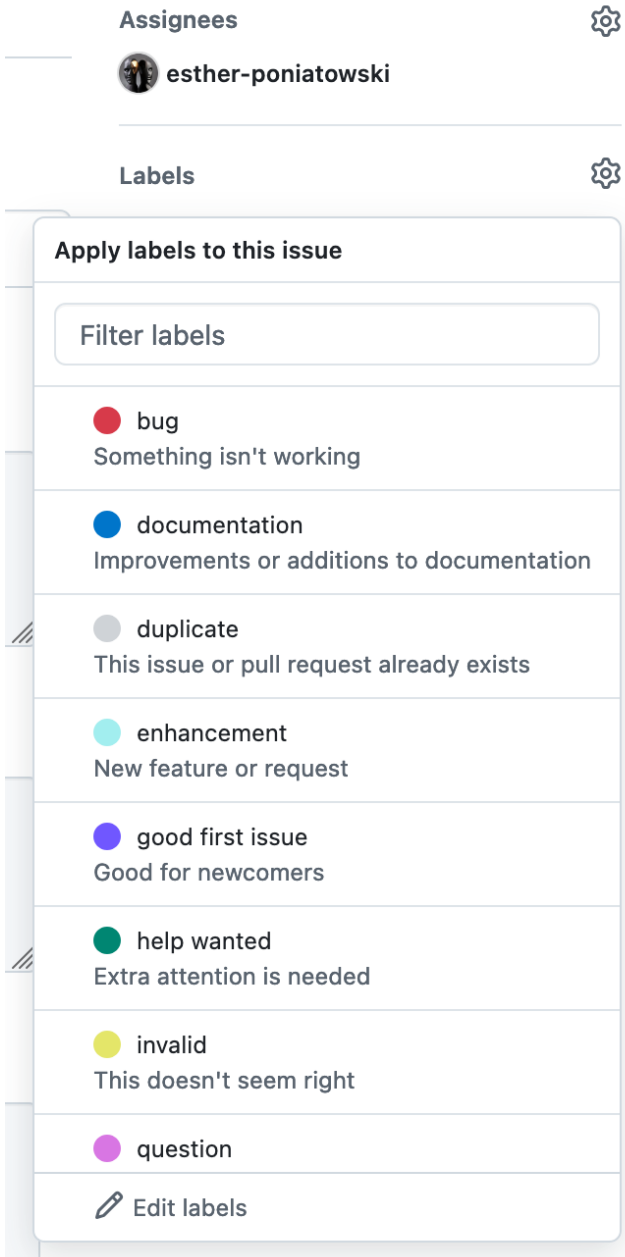
Beta You're using an issue form, a new type of issue template.

Create a Repository

Establish an Issue Template

Recommendation

Define labels to match commit prefixes



Create a Repository

Practical Application

Issue: ProjectTemplate Issue Form

General template for new issues. If this doesn't look right, choose a different type.

Add a title

Set up the directory structurechore

Scope

Related branch, files, code lines...

Write | Preview

All

Markdown is supported | Paste, drop, or click to add files

Assignees

esther-poniatowski

Labels

Apply labels to this issue

chore

chore
Miscellaneous: manage dependencies, confi...

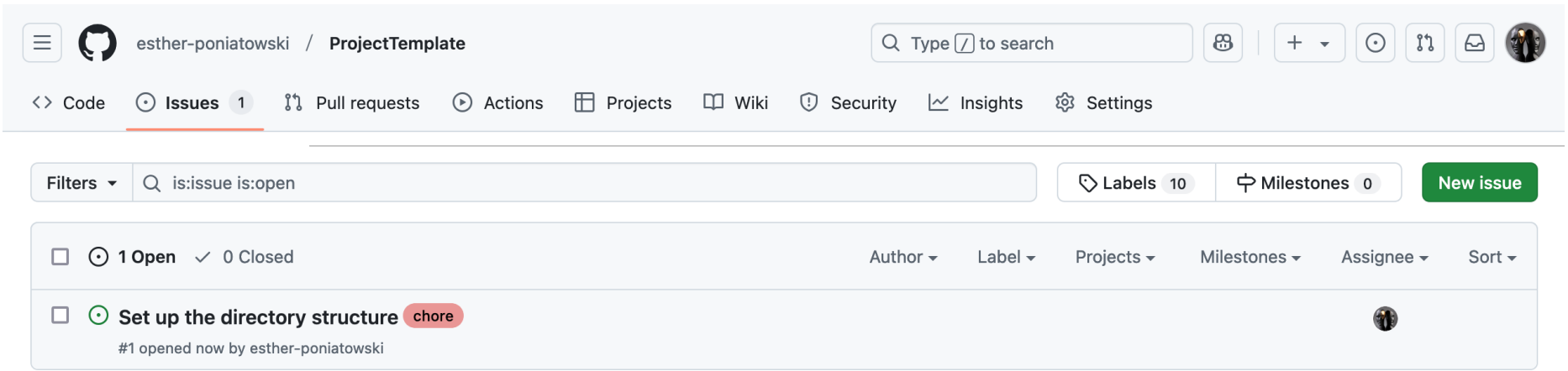
duplicate
This issue or pull request already exists

Edit labels

Shows branches and pull requests linked to this issue.

Helpful resources

Create a Repository



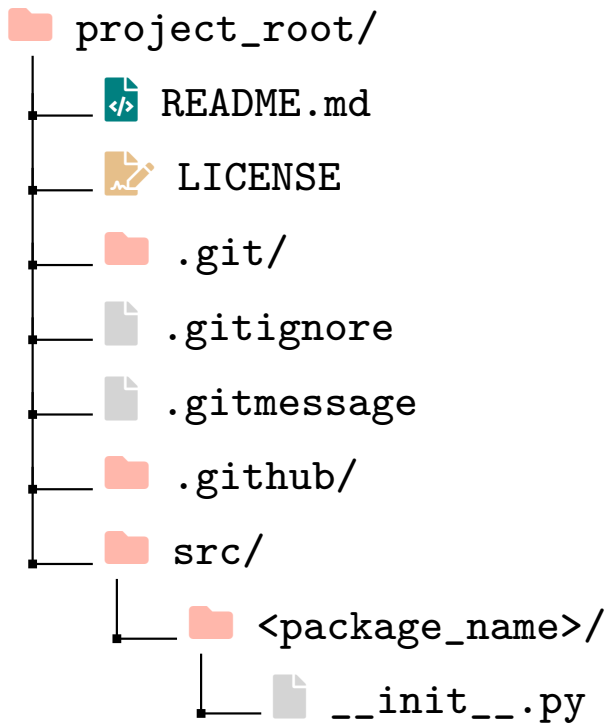
Initialization Workflow

Initialization **Development** **Deployment**

- Create Remote and Local Repositories
- Establish Rules for the Project
- **Construct the Project Structure**
- Create a Virtual Environment
- Create a Code Workspace

Create a Repository

Construct the Skeleton of the Project



Basic Directory Structure

`src` (source code)
containing packages

Create a Repository

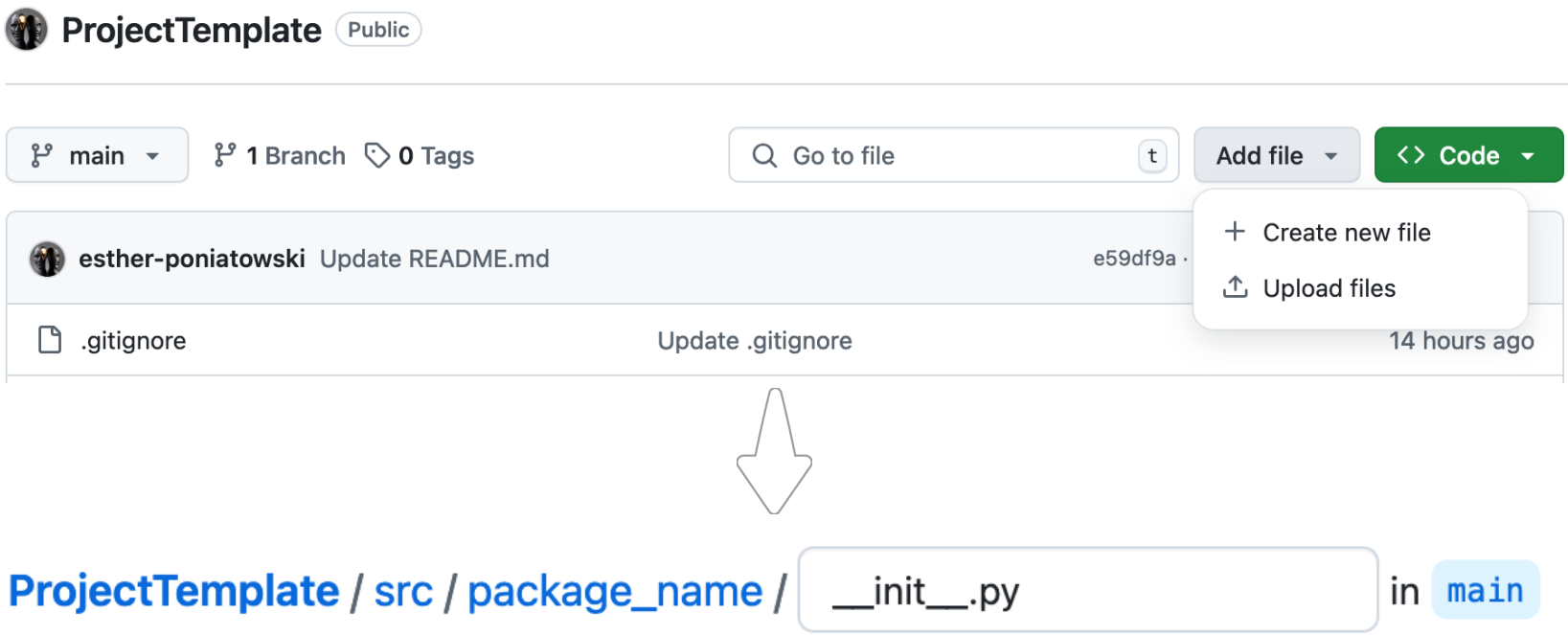
Construct the Skeleton of the Project

`__init__.py`

Tool	Python 
Type	Python Script
Location	At the root of any package (i.e. directory containing modules)
Content	**Package declaration** Package-level documentation (optional) Convenience imports (optional) Namespace (optional) Package Configuration (optional)

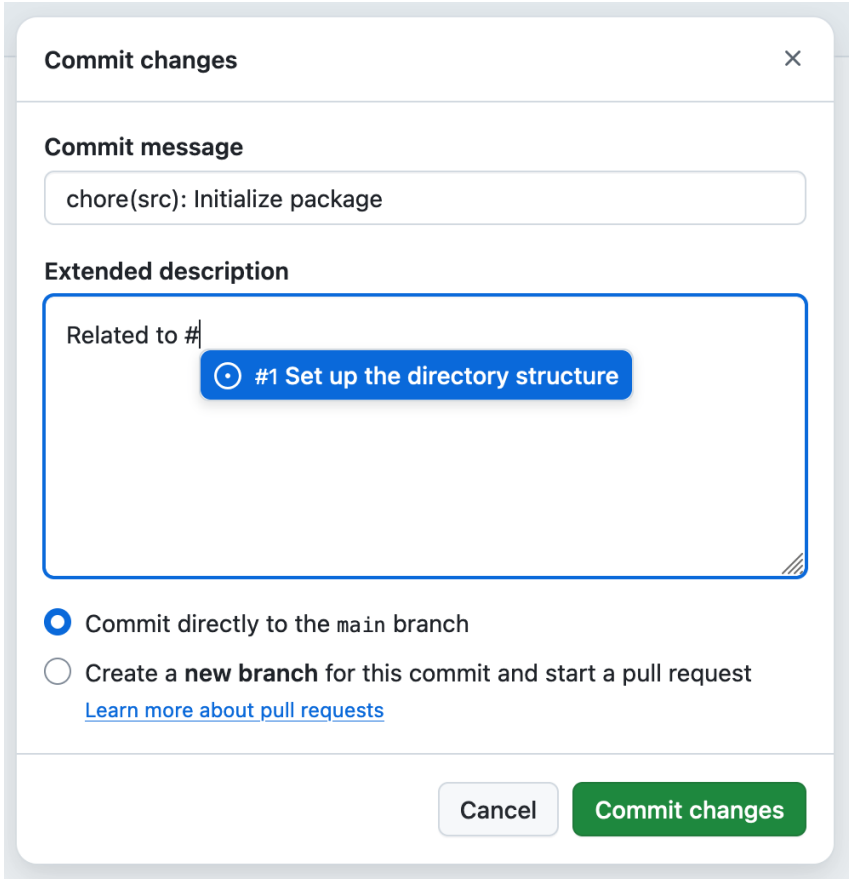
Create a Repository

Construct the Skeleton of the Project



Create a Repository

Construct the Skeleton of the Project



Initialization Workflow

Initialization **Development** **Deployment**

- Create Remote and Local Repositories
- Establish Rules for the Project
- Construct the Project Structure
- **Create a Virtual Environment**
- Create a Code Workspace

Initialization Workflow

Create a Virtual Environment

Goal	Specify dependencies to ensure versions, Isolate dependencies to avoid conflicts.
Importance	Reliability, Reproducibility
Tools	Conda

Create a Virtual Environment

Warning

Do not install packages globally (i.e. in system-wide Python).

↪ Risk: Destabilize the base installation required for conda to operate.

Recommendation

Use virtual environments to isolate dependencies for each project.

Create a Virtual Environment

Comparison: Conda / pip + venv

	Conda	**pip + venv**
Programming Languages	Multiple languages (Python, R, C++...)	Python-specific
Environment Location	Centralized, named folders	Local to project
Dependency Resolution	Automatic	Manual
Package Source	Conda channels	PyPI
Pre-compiled Binaries	Often available	Limited availability
Cross-platform Compatibility	High	Variable

Create a Virtual Environment

Warning

Mixing pip and Conda can lead to issues

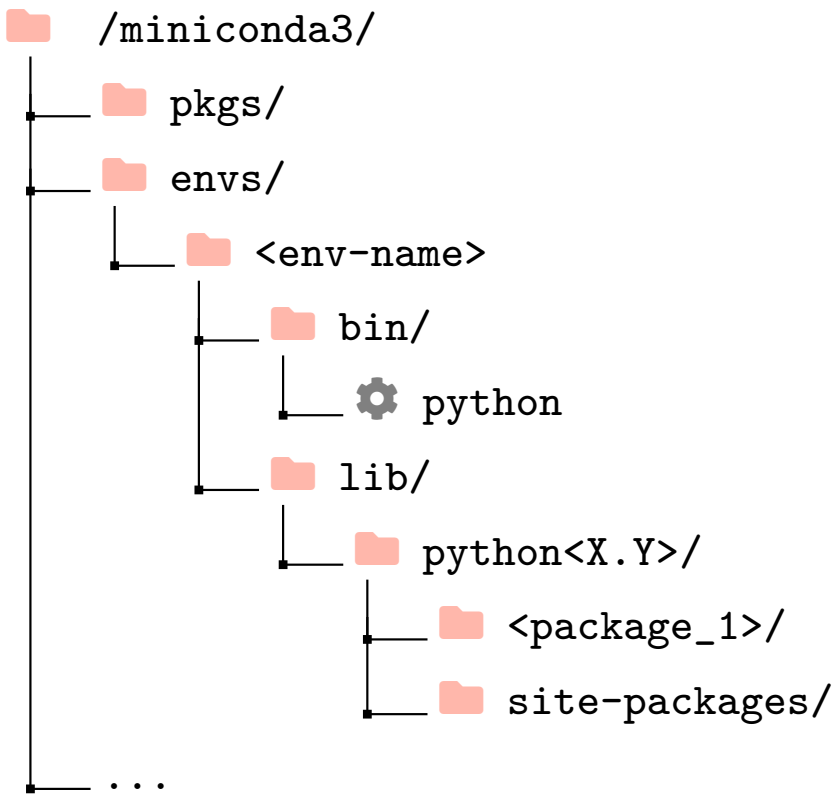
- Changes made by pip can create conflicts or broken environments.
- When pip is used independently (outside of a conda environment), it installs packages globally or in a user-specific directory, which may not be accessible within the conda environment.

Recommendation

- Use Conda when possible.
- Use pip **within the activated conda environment** only for packages that are not available through Conda channels.

Create a Virtual Environment

Structure of Conda Environments



Cache storing pre-compiled packages

Centralized registry of environments

Environment-specific directory

Executables: Python, pip, jupyter...

Python interpreter

Python version-specific resources

Standard library package (example)

Installed third-party packages

Create a Virtual Environment

Activating a Conda Environment:

```
$ conda activate <env-name>
(env-name) $
```

Outcomes

Update of Environment Variables

- `PATH` (system var.): Prepend the environment's `bin` directory.

  ↪ *Environment-specific* executables are prioritized over the system's, including `python` (Python interpreter)

- `sys.path` (Python var.): Include the environment's `site-packages`.

  ↪ *Environment-specific* packages are accessible from Python scripts.

Create a Virtual Environment

Define the Dependencies

environment.yml

Tool	Conda
Type	YAML
Location	./ (project root)
Content	List of dependencies for the virtual environment.

Create a Virtual Environment

Define the Dependencies

```yaml
name: <env-name>
channels:
    - conda-forge
    - defaults
dependencies:
    - python
    - numpy
    - pandas
    - scipy
    - matplotlib
    - pip:
        - seaborn==0.11.1
```

Create a Virtual Environment

Install the Dependencies

Create a new Conda environment using the `environment.yml` file:

```
$ conda env create -f environment.yml
```

Outcomes

- New Conda environment with all the dependencies installed.
- Name: Specified in the `environment.yml` file.
- Location of the Conda directory: `envs/`.

Create a Virtual Environment

Update the Dependencies

Recommendation

To add or remove dependencies during the project's lifecycle:

- Update the `environment.yml` file.
- Update the Conda environment from the file.

```
$ conda env update --file environment.yml --prune
```

Option `--prune`: Remove dependencies that are no longer required.

Create a Virtual Environment

Install the Work-In-Progress Package in Development Mode

Register the package in "editable mode":

```
$ conda activate <env-name>
$ pip install -e /src/<package-name>
```

Outcomes

Add a link to the source code directory in the `site-packages` directory of the Python interpreter of the activated Conda environment.

- ▷ Enable the package to be imported when the environment is active.
- ▷ Avoid re-installing the package after each modification.
- ▷ Test the package in the development environment.