Technical Implementation Overview: Quarto Docs Workflow with GitHub Actions

Author: **European Environment Agency (EEA)**

Date: **Error! Unknown document property name.**

Version: **Error! Unknown document property name.**

Index

1 Overview....1
2 Branches & Workflow....1
2.1 Branching Strategy....1
3 Publishing via GitHub Actions....1
3.1 Implementation Details....1
4 Automatic Tagging (Versioning)....2
4.1 How It's Done....2
4.2 Benefits....2
5 Optimization Recommendations....2
5.1 Resource Optimization....2
5.2 Workflow Simplicity....2

1 Overview

This document covers the technical details and strategy used to manage Quarto (.qmd) documents within our project. It describes the workflow, publishing steps, automatic version tagging, and includes optimization tips.

2 Branches & Workflow

We have three main branches:

- **main:** Stable, production-ready content.
- **test:** A shared environment for testing and reviewing changes.
- **Feature branches:** Individual branches based on `test` for user-specific changes.

2.1 Branching Strategy

- Users create feature branches from the `test` branch for independent work.
- Changes are reviewed and merged first into `test`, then after approval, merged into `main`.

3 Publishing via GitHub Actions

3.1 Implementation Details

GitHub Actions automates the workflow:

- **Rendering:** Documents are rendered from `.qmd` files to HTML and DOCX format using Quarto.
- **Conversion:** LibreOffice macros automatically convert DOCX files into PDFs.
- **Deployment:** Rendered PDFs and HTML files are deployed to github-pages. The `main` and `test` branches each have their own GitHub Pages environments—the official pages (from `main`) and a separate environment for testing (`test`).

4 Automatic Tagging (Versioning)

4.1 How It's Done

Automatic tagging occurs on each merge to the `main` branch. The GitHub Action (`mathieudutour/github-tag-action`) creates semantic tags automatically.

Configuration:

```
- name: Auto Tagging
  uses: mathieudutour/github-tag-action@v6.2
  with:
    github_token: ${{ secrets.GITHUB_TOKEN }}
```

4.2 Benefits

- Removes the need for manual tagging.
- Provides clear, consistent versioning.
- Makes reverting or referencing historical versions straightforward.

5 Optimization Recommendations

5.1 Resource Optimization

- Maintain one common test environment rather than multiple per-branch environments to minimize complexity and costs.

5.2 Workflow Simplicity

- Simple branching strategy (just `main`, `test`, and feature branches) to avoid confusion and ease maintenance.