Publishing

Session 6

Mickaël CANOUIL, Ph.D.[1*]

Saturday, the 7th of February, 2026

Affiliations

[1] Independent Contractor, Lille, France

ORCID

Mickaël CANOUIL, Ph.D.: 0000-0002-3396-4549

[*]Corresponding author: pro@mickael.canouil.dev

Table of Contents

Table of Contents

Table of Contents

1. Session Overview

1.1. Session 6: Publishing

- Publishing Options Overview
  - Comparison of Quarto Pub, GitHub Pages, Netlify, and other hosting services.
  - Understanding benefits and limitations of each platform.
  - Choosing the right publishing method for different use cases.
- Quarto Pub and GitHub Pages
  - Step-by-step Quarto Pub deployment workflow.
  - Three methods for GitHub Pages: docs folder, quarto publish, and GitHub Actions.
  - Understanding `_publish.yml` and repository configuration.
- Code Execution Strategies
  - Local execution with freeze vs CI execution.
  - Environment management and dependency handling.
  - Version control best practices for published projects.
- Advanced Topics
  - Custom domains configuration.
  - Automated deployment workflows.
  - Troubleshooting common publishing issues.

1.2. Session Objectives

This session explores comprehensive publishing strategies for Quarto projects, covering multiple deployment platforms, automation workflows, and best practices for different use cases.

1.3. Learning Objectives

By the end of this session, participants will be able to:

- Choose the right publishing method for your use case.
- Deploy a Quarto project to multiple platforms.
- Set up automated publishing workflows.
- Troubleshoot common publishing issues.
- Configure custom domains and advanced features.
- Distribute extensions via GitHub or archives.

2. Quarto Publishing

2.1. Publishing Options Overview

Service	Best For	Key Features
Quarto Pub	Public content, getting started	Free, simple, 100MB limit
GitHub Pages	Open source projects	Git integration, custom domains
Netlify	Professional sites	Advanced features, previews
Posit Connect	Enterprise/organisations	Security, authentication
Other Services	Custom requirements	Firebase, S3, self-hosted

Rule of thumb

Start with Quarto Pub for learning, use GitHub Pages for projects, Netlify for production sites

3. Quarto Pub

3.1. Your First Quarto Pub Deployment

1. Create an account.
   Visit quartopub.com and sign up.
2. Publish from CLI.

```bash
quarto publish quarto-pub
```

3. Authenticate.
   - Browser opens for authorization
   - Confirm publishing permissions
   - Site deploys automatically!

> **i** That's it!
>
> Your site is live at `https://username.quarto.pub/project-name`.

3.2. Understanding `_publish.yml`

Created automatically when you first publish:

```yaml
# _publish.yml
- source: project
  quarto-pub:
    - id: "abc123-def456-ghi789"
      url: "https://myusername.quarto.pub/myproject"
```

- Safe for version control (no credentials).
- Shareable across team members.
- Service configuration stored locally.
- Reusable for subsequent deployments.

4. Hands-On Exercise: Quarto Pub

4.1. Exercise Overview

Objective: Publish the branded project you created in Session 5 using Quarto Pub.

Example Code (reused from Session 5): Exercises

```bash
tar -xzf "05b-exercises.tar.gz" -C "06-publishing"
```

4.2. Publishing to Quarto Pub

1. Create a simple Quarto document.
2. Add some content (text, code, plots).
3. Run `quarto publish quarto-pub`.
4. Share your live URL with a neighbour.

```bash
# Publish without prompts
quarto publish quarto-pub --no-prompt

# Publish without opening browser
quarto publish quarto-pub --no-browser

# Publish without re-rendering
quarto publish quarto-pub --no-render
```

4.3. Success Criteria

☑ You've successfully completed the exercise if you can:

- Successfully publish your Quarto project to Quarto Pub.

5. GitHub Pages

5.1. Three Ways to Use GitHub Pages

5.1.1. Render to `docs/`

- Simplest approach
- Commit rendered output
- Good for small projects

5.1.2. `quarto publish`

- Local rendering
- Automatic deployment
- Clean separation

5.1.3. GitHub Actions

- Fully automated
- Triggered by commits
- Most professional

5.2. Method 1: Render to `docs/`

- Configure Output Directory

```yaml
# _quarto.yml
project:
  type: website
  output-dir: docs
```

- Add `.nojekyll` File

```bash
touch .nojekyll
```

- Render and Push

```bash
quarto render
git add .
git commit -m "feat: Add rendered site"
git push
```

- Configure Repository

  Settings ⟶ Pages ⟶ Source: Deploy from branch ⟶ Branch: main ⟶ Folder: /docs

5.3. Method 2: `quarto publish gh-pages`

- Setup Source Branch

```bash
# Create gh-pages branch
git checkout --orphan gh-pages
git rm -rf .
git commit --allow-empty -m "chore: Initial gh-pages commit"
```

```
git push origin gh-pages
git checkout main
```

- Configure .gitignore

```txt
_site/
_book/
```

- Publish

```bash
quarto publish gh-pages
```

5.4. Method 3: GitHub Actions Workflow

- Create Workflow File: `.github/workflows/publish.yml`.

```yaml
name: Quarto Publish
on:
  workflow_dispatch:
  push:
    branches: main

jobs:
  build-deploy:
    runs-on: ubuntu-latest
    permissions:
      contents: write
    steps:
      - name: Check out repository
        uses: actions/checkout@v4

      - name: Set up Quarto
        uses: quarto-dev/quarto-actions/setup@v2

      - name: Render and Publish
        uses: quarto-dev/quarto-actions/publish@v2
        with:
          target: gh-pages
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

6. Hands-On Exercise: GitHub Pages

6.1. Exercise Overview

Objective: Publish the branded project you created in Session 5 using GitHub Pages.

Example Code (reused from Session 5): Exercises

```bash
tar -xzf "05b-exercises.tar.gz" -C "06-publishing"
```

6.2. Publishing to GitHub Pages

Choose one method and deploy your project:

6.2.1. Method 1: docs/ folder

1. Modify `_quarto.yml`.
2. Add `.nojekyll`.
3. Render and push.
4. Configure in GitHub settings.

6.2.2. Method 2: quarto publish

1. Setup gh-pages branch.
2. Update `.gitignore`.
3. Run `quarto publish gh-pages`.
4. Verify deployment.

6.3. Success Criteria

☑ You've successfully completed the exercise if you can:

- Successfully publish your Quarto project to GitHub Pages using one of the methods.

7. Code Execution Strategies

7.1. The Big Decision: Where to Execute Code?

7.1.1. Local Execution + Freeze

_quarto.yml

```yaml
execute:
  freeze: auto
```

Pros:

- Faster CI builds.
- Consistent environment.
- Handle legacy code.

Cons:

- Manual re-execution needed.
- Larger repository size.

7.1.2. CI Execution

Pros:

- Always fresh execution.
- Smaller repositories.
- Reproducible environments.

Cons:

- Longer build times.
- Dependency management.
- Potential CI failures.

7.2. Freeze Strategy Implementation

- Enable Freezing

_quarto.yml

```yaml
execute:
  freeze: auto
```

- Re-render Locally

BASH

```bash
quarto render
```

- Version Control

BASH

```bash
git add _freeze/
git commit -m "chore: Update frozen computations"
git push
```

> ! Important
>
> The `_freeze/` directory stores computational results and must be committed to Git.

7.3. CI Execution with Dependencies

7.3.1. Python/Jupyter Example

```yaml
- name: Install Python and Dependencies
  uses: actions/setup-python@v5
  with:
    python-version: '3.13'
    cache: 'pip'

- name: Install Jupyter
  run: pip install jupyter

- name: Install Requirements
  run: pip install -r requirements.txt
```

7.3.2. R/Knitr Example

```yaml
- name: Install R
  uses: r-lib/actions/setup-r@v2
  with:
    r-version: '4.5.0'

- name: Install R Dependencies
  uses: r-lib/actions/setup-renv@v2
  with:
    cache-version: 1
```

7.4. Version Control Strategies

Use a freeze strategy (recommended):

```yaml
execute:
  freeze: auto
```

Commit to Git:

- Source files (`.qmd`).
- `_freeze/` directory.
- Configuration files.

Don't Commit:

- `_site/` , `_book/` , or `_manuscript/` .
- Temporary files.
- Local dependencies.

7.5. Environment Management

Use a tool to manage your R , Python , or Julia environment:

- `renv` for R.
- `uv` , `pip` , or `conda` for Python.
- `Pkg` for Julia.

8. Advanced Topics

8.1. Custom Domains

8.1.1. GitHub Pages Custom Domain

1. Add `CNAME` file to project root:

```txt
yourdomain.com
```

2. Configure DNS records:
   - Apex domain: A records to GitHub IPs.
   - Subdomain: CNAME to `username.github.io`.
3. Enable in repository settings.

8.1.2. Netlify Custom Domain

- Automatic HTTPS.
- Easy DNS management.
- Domain purchasing available.

9. Extension Distribution

9.1. Distribution Methods

9.1.1. GitHub Distribution (Recommended)

```bash
# Users install with:
quarto add yourorg/company-report

# Or specific version/tag:
quarto add yourorg/company-report@v1.0.0

# Or from branch:
quarto add yourorg/company-report@main
```

Benefits: Version control, easy updates, namespace management (`yourorg/extension`).

9.1.2. Archive Distribution

```bash
# From URL:
quarto add https://example.com/company-report.zip

# From local file:
quarto add ./company-report.tar.gz
```

Benefits: No GitHub required, works offline, controlled distribution.

9.2. Publishing Best Practices

Prepare your extension:

1. Add README.md - Installation instructions and documentation.
2. Include LICENSE - Specify usage terms.
3. Create example.qmd - Demonstrate usage.
4. Version with tags - Use semantic versioning (e.g., `v1.0.0`).

> **Tip**
>
> See Distributing Extensions for complete details.

