

Publishing

Session 6

Mickaël CANOUIL, *Ph.D.* 

pro@mickael.canouil.dev

Independent Contractor

Saturday, the 7th of February, 2026

- Session 6: Publishing
- Session Objectives
- Learning Objectives

Session Overview

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.

- **Code Execution Strategies**

- Local execution with freeze vs CI execution.
- Environment management and dependency handling.
- Version control best practices for published projects.

- **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.

- **Advanced Topics**

- Custom domains configuration.
- Automated deployment workflows.
- Troubleshooting common publishing issues.

Session Objectives

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

Learning Objectives

Learning Objectives

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

Learning Objectives

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

- Choose the right publishing method for your use case.

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.

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.

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.

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.

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.

- Publishing Options Overview

Quarto Publishing

Publishing Options Overview

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

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

- Your First Quarto Pub Deployment
- Understanding `_publish.yml`

Quarto Pub

Your First Quarto Pub Deployment

Your First Quarto Pub Deployment

1. Create an account.

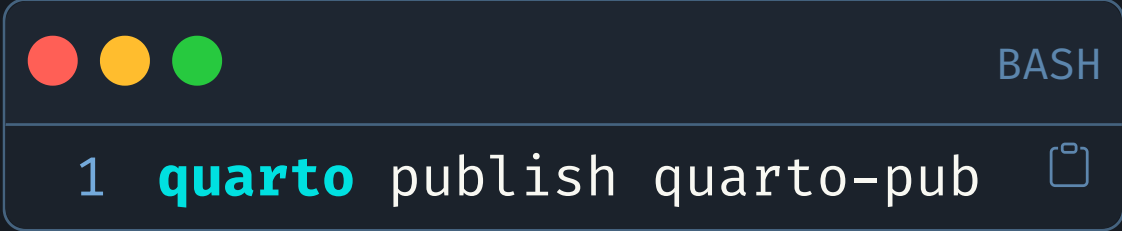
Visit quartopub.com and sign up.

Your First Quarto Pub Deployment

1. Create an account.

Visit quartopub.com and sign up.

2. Publish from CLI.



```
BASH
1 quarto publish quarto-pub
```

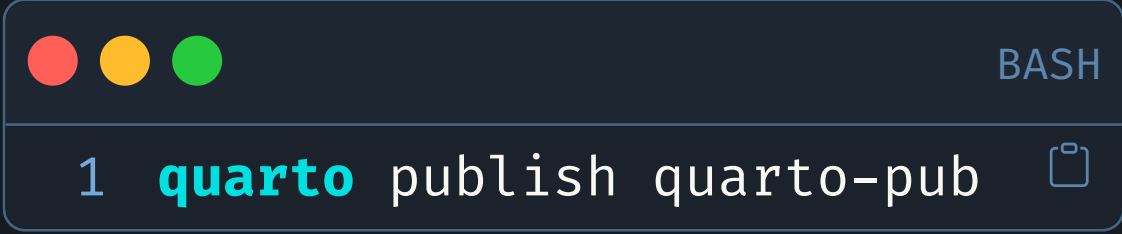
A terminal window with a dark background. The title bar shows three colored circles (red, yellow, green) and the text 'BASH'. The command prompt shows '1 quarto publish quarto-pub' with a clipboard icon to the right.

Your First Quarto Pub Deployment

1. Create an account.

Visit quartopub.com and sign up.

2. Publish from CLI.



```
BASH
1 quarto publish quarto-pub
```

A terminal window with a dark background. The title bar shows three colored circles (red, yellow, green) and the text 'BASH'. The command prompt shows '1 quarto publish quarto-pub' with a copy icon to the right.

3. Authenticate.

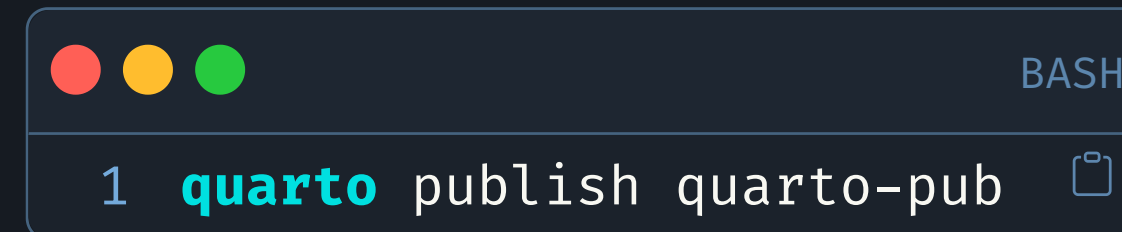
- Browser opens for authorization
- Confirm publishing permissions
- Site deploys automatically!

Your First Quarto Pub Deployment

1. Create an account.

Visit quartopub.com and sign up.

2. Publish from CLI.



```
BASH
1 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`.

Understanding `_publish.yml`

Understanding `_publish.yml`

Created automatically when you first publish:

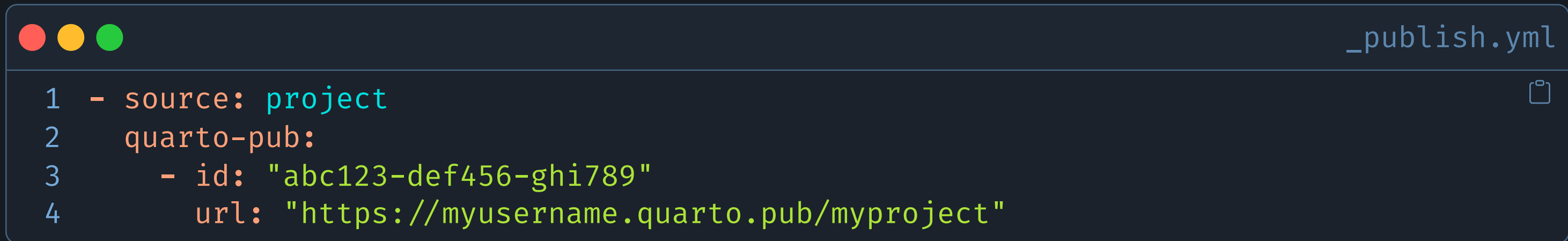


```
_publish.yml
1 - source: project
2   quarto-pub:
3     - id: "abc123-def456-ghi789"
4       url: "https://myusername.quarto.pub/myproject"
```



Understanding `_publish.yml`

Created automatically when you first publish:

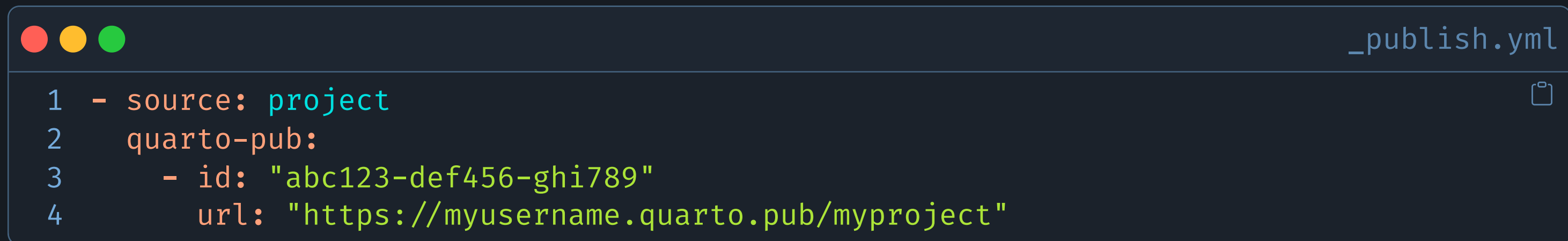
A terminal window with a dark background and light blue text. The title bar shows three colored circles (red, yellow, green) on the left and the filename "_publish.yml" on the right. The content of the file is displayed on four lines, numbered 1 to 4 on the left. The text is color-coded: red for hyphens, blue for keys, green for values, and yellow for strings.

```
1 - source: project
2   quarto-pub:
3     - id: "abc123-def456-ghi789"
4       url: "https://myusername.quarto.pub/myproject"
```

- **Safe for version control** (no credentials).

Understanding `_publish.yml`

Created automatically when you first publish:

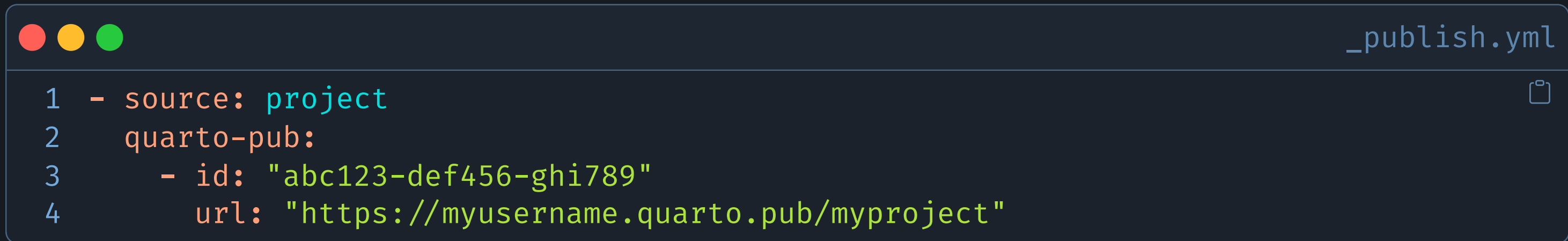
A terminal window with a dark background and light blue text. The title bar shows three colored circles (red, yellow, green) on the left and the filename "_publish.yml" on the right. The content of the file is a YAML configuration with four lines, numbered 1 to 4 on the left. Line 1: "- source: project". Line 2: " quarto-pub:". Line 3: " - id: 'abc123-def456-ghi789'". Line 4: " url: 'https://myusername.quarto.pub/myproject'". A small clipboard icon is visible on the right side of the terminal window.

```
1 - source: project
2   quarto-pub:
3     - id: "abc123-def456-ghi789"
4       url: "https://myusername.quarto.pub/myproject"
```

- **Safe for version control** (no credentials).
- **Shareable** across team members.

Understanding `_publish.yml`

Created automatically when you first publish:

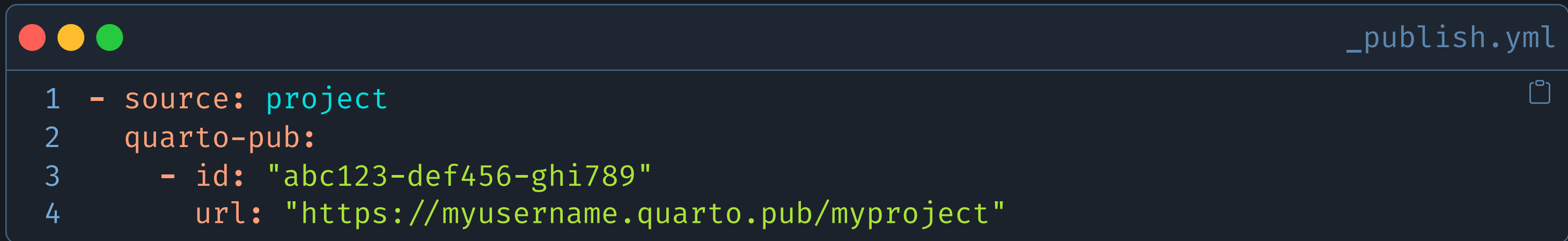
A terminal window with a dark background and light blue text. The title bar shows three colored circles (red, yellow, green) on the left and the filename "_publish.yml" on the right. The content of the file is displayed as a YAML configuration with line numbers 1 through 4 on the left. The configuration defines the source as 'project', the publisher as 'quarto-pub', and a single publication entry with an 'id' and a 'url'.

```
1 - source: project
2   quarto-pub:
3     - id: "abc123-def456-ghi789"
4       url: "https://myusername.quarto.pub/myproject"
```

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

Understanding `_publish.yml`

Created automatically when you first publish:

A terminal window with a dark background and light blue text. The title bar shows three colored circles (red, yellow, green) on the left and the filename "_publish.yml" on the right. The content of the file is displayed as a YAML configuration with line numbers 1 through 4 on the left. The configuration defines a source as 'project' and a quarto-pub entry with an id and a url.

```
1 - source: project
2   quarto-pub:
3     - id: "abc123-def456-ghi789"
4       url: "https://myusername.quarto.pub/myproject"
```

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

- Exercise Overview
- Publishing to Quarto Pub
- Success Criteria

Hands-On Exercise: Quarto Pub

Exercise Overview

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

Example Code (reused from Session 5):  [Exercises](#)

BASH

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



Publishing to Quarto Pub

Publishing to Quarto Pub

1. Create a simple Quarto document.

Publishing to Quarto Pub

1. Create a simple Quarto document.
2. Add some content (text, code, plots).

Publishing to Quarto Pub

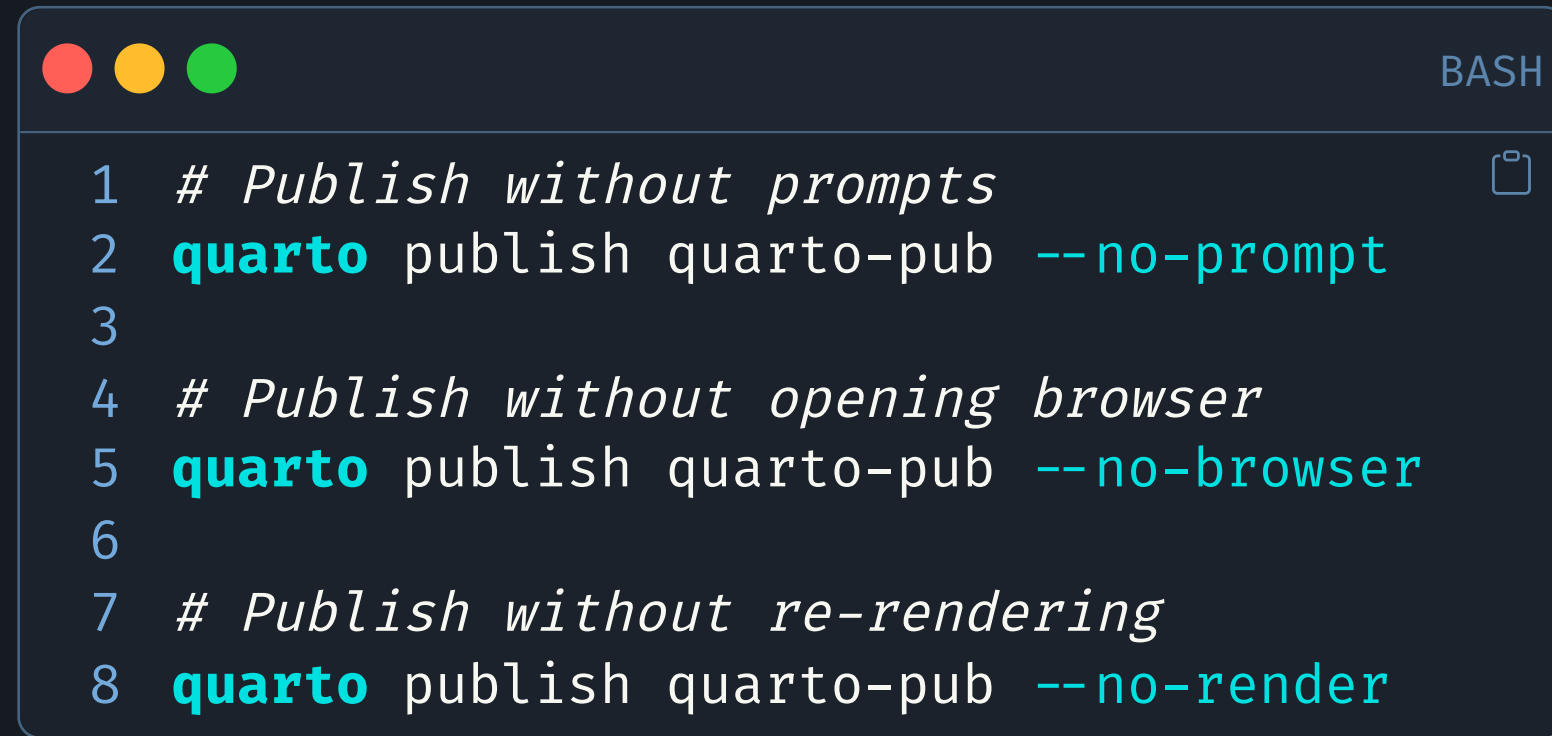
1. Create a simple Quarto document.
2. Add some content (text, code, plots).
3. Run `quarto publish quarto-pub`.

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.

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.

A terminal window with a dark background and light blue text. The title bar shows three colored circles (red, yellow, green) and the text 'BASH'. The terminal content consists of eight lines of code, with line numbers 1 through 8 on the left. The code includes comments and Quarto publish commands with various flags. A clipboard icon is visible in the top right corner of the terminal area.

```
1 # Publish without prompts
2 quarto publish quarto-pub --no-prompt
3
4 # Publish without opening browser
5 quarto publish quarto-pub --no-browser
6
7 # Publish without re-rendering
8 quarto publish quarto-pub --no-render
```

Success Criteria

You've successfully completed the exercise if you can:

- Successfully publish your Quarto project to Quarto Pub.

- Three Ways to Use GitHub Pages
- Method 1: Render to docs/
- Method 2: quarto publish gh-pages
- Method 3: GitHub Actions Workflow

GitHub Pages

Three Ways to Use GitHub Pages

Three Ways to Use GitHub Pages

Render to docs/

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

Three Ways to Use GitHub Pages

Render to docs/

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

quarto publish

- Local rendering
- Automatic deployment
- Clean separation

Three Ways to Use GitHub Pages

Render to docs/

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

quarto publish

- Local rendering
- Automatic deployment
- Clean separation

GitHub Actions

- Fully automated
- Triggered by commits
- Most professional

Method 1: Render to docs/

Method 1: Render to docs/

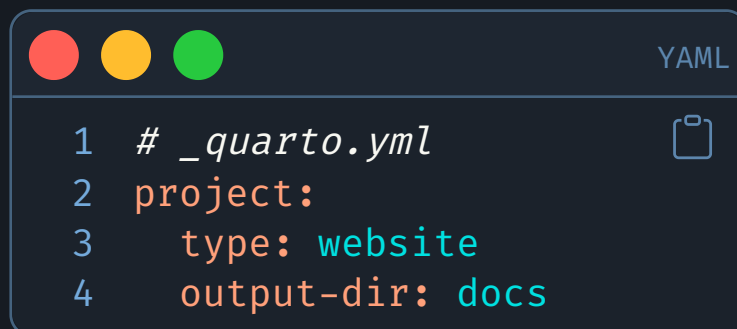
- Configure Output Directory



```
1 # _quarto.yml
2 project:
3   type: website
4   output-dir: docs
```

Method 1: Render to docs/

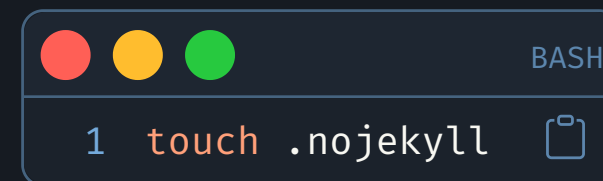
- Configure Output Directory



```
1 # _quarto.yml
2 project:
3   type: website
4   output-dir: docs
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left. The title bar on the right says "YAML". The text is a YAML configuration for a Quarto project, with line numbers 1 through 4 on the left. A clipboard icon is in the top right corner.

- Add .nojekyll File



```
1 touch .nojekyll
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left. The title bar on the right says "BASH". The text shows a command to create a file, with line number 1 on the left. A clipboard icon is in the top right corner.

Method 1: Render to docs/

- Configure Output Directory

```
1 # _quarto.yml
2 project:
3   type: website
4   output-dir: docs
```

- Add .nojekyll File

```
1 touch .nojekyll
```

- Render and Push

```
1 quarto render
2 git add .
3 git commit -m "feat: Add rendered site"
4 git push
```

Method 1: Render to docs/

- Configure Output Directory

```
1 # _quarto.yml
2 project:
3   type: website
4   output-dir: docs
```

- Render and Push

```
1 quarto render
2 git add .
3 git commit -m "feat: Add rendered site"
4 git push
```

- Add .nojekyll File

```
1 touch .nojekyll
```

- Configure Repository

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

Method 2: `quarto publish gh-pages`

Method 2: quarto publish gh-pages

- Setup Source Branch



```
1 # Create gh-pages branch
2 git checkout --orphan gh-pages
3 git rm -rf .
4 git commit --allow-empty -m "chore: Initial gh-pages commit"
5 git push origin gh-pages
6 git checkout main
```

A terminal window with a dark background and light blue text. The window has three colored window control buttons (red, yellow, green) in the top left corner. The title bar on the right says "BASH". The terminal content shows a sequence of six numbered git commands to create and push a new branch named 'gh-pages'.

Method 2: quarto publish gh-pages

- Setup Source Branch

```
BASH
1 # Create gh-pages branch
2 git checkout --orphan gh-pages
3 git rm -rf .
4 git commit --allow-empty -m "chore: Initial gh-pages commit"
5 git push origin gh-pages
6 git checkout main
```

- Configure .gitignore

```
TXT
1 _site/
2 _book/
```

Method 2: `quarto publish gh-pages`

- Setup Source Branch

```
BASH
1 # Create gh-pages branch
2 git checkout --orphan gh-pages
3 git rm -rf .
4 git commit --allow-empty -m "chore: Initial gh-pages commit"
5 git push origin gh-pages
6 git checkout main
```

- Configure `.gitignore`

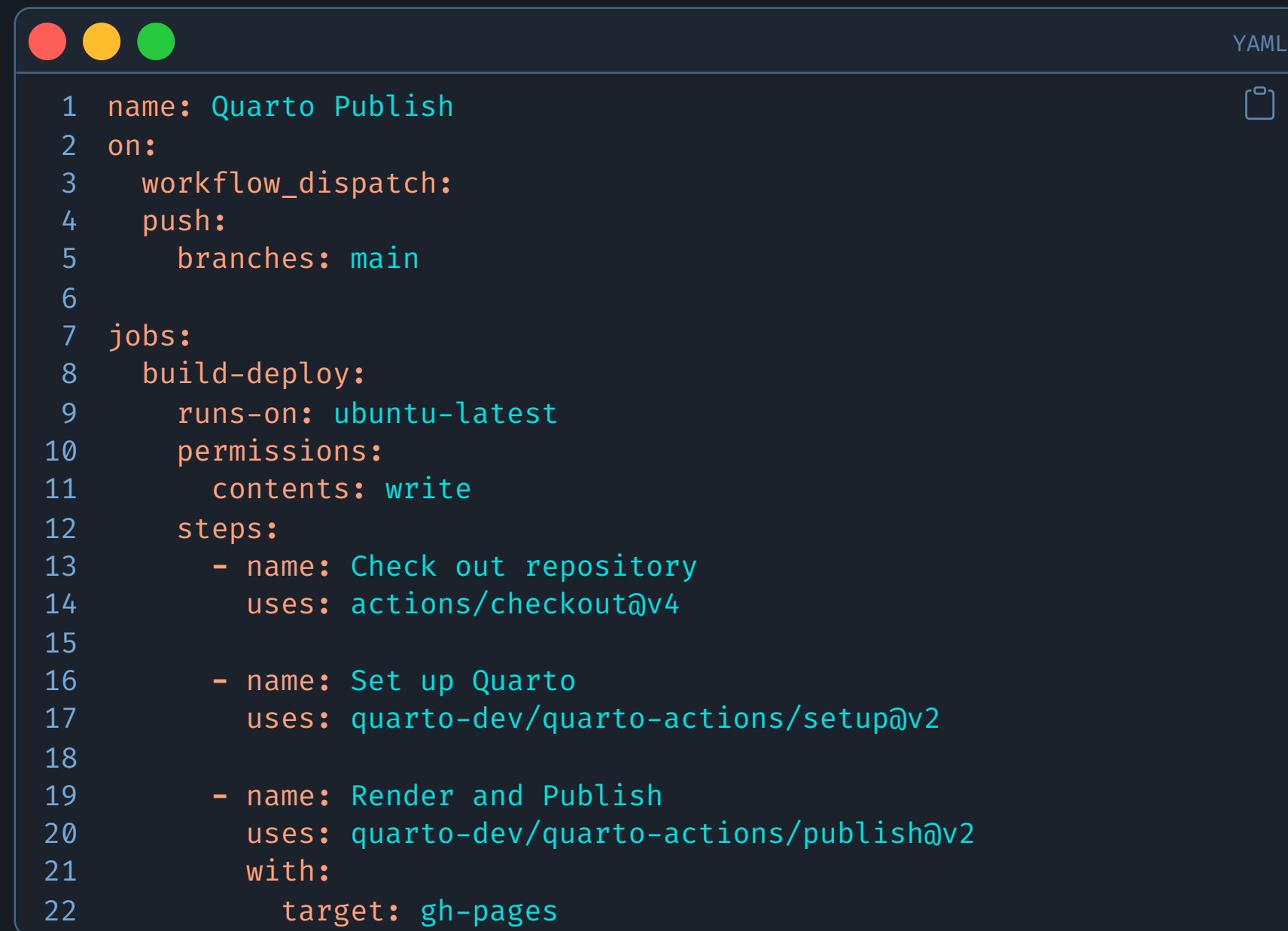
```
TXT
1 _site/
2 _book/
```

- Publish

```
BASH
1 quarto publish gh-pages
```

Method 3: GitHub Actions Workflow

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



```
1 name: Quarto Publish
2 on:
3   workflow_dispatch:
4   push:
5     branches: main
6
7 jobs:
8   build-deploy:
9     runs-on: ubuntu-latest
10    permissions:
11      contents: write
12    steps:
13      - name: Check out repository
14        uses: actions/checkout@v4
15
16      - name: Set up Quarto
17        uses: quarto-dev/quarto-actions/setup@v2
18
19      - name: Render and Publish
20        uses: quarto-dev/quarto-actions/publish@v2
21        with:
22          target: gh-pages
```

- Exercise Overview
- Publishing to GitHub Pages
- Success Criteria

Hands-On Exercise: GitHub Pages

Exercise Overview

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

Example Code (reused from Session 5):  [Exercises](#)

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

BASH

Publishing to GitHub Pages

Choose **one** method and deploy your project:

Publishing to GitHub Pages

Choose **one** method and deploy your project:

Method 1: docs/ folder

Publishing to GitHub Pages

Choose **one** method and deploy your project:

Method 1: docs/ folder

Method 2: quarto publish

Publishing to GitHub Pages

Choose **one** method and deploy your project:

Method 1: docs/ folder

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

Method 2: quarto publish

Publishing to GitHub Pages

Choose **one** method and deploy your project:

Method 1: docs/ folder

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

Method 2: quarto publish

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

Success Criteria

You've successfully completed the exercise if you can:

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

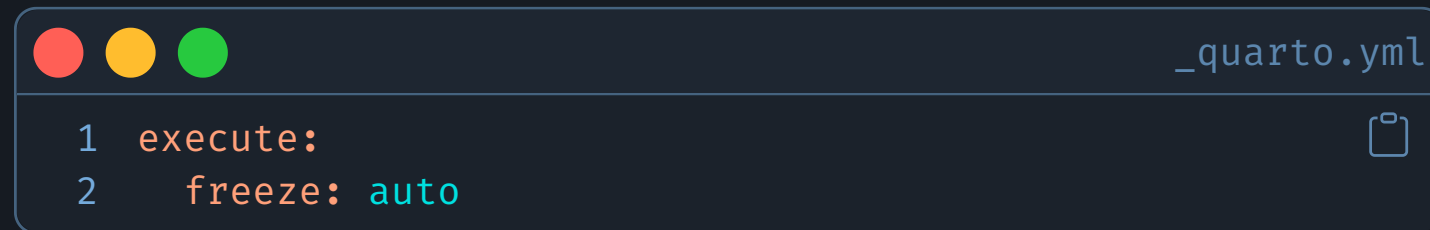
- The Big Decision: Where to Execute Code?
- Freeze Strategy Implementation
- CI Execution with Dependencies
- Version Control Strategies
- Environment Management

Code Execution Strategies

The Big Decision: Where to Execute Code?

The Big Decision: Where to Execute Code?

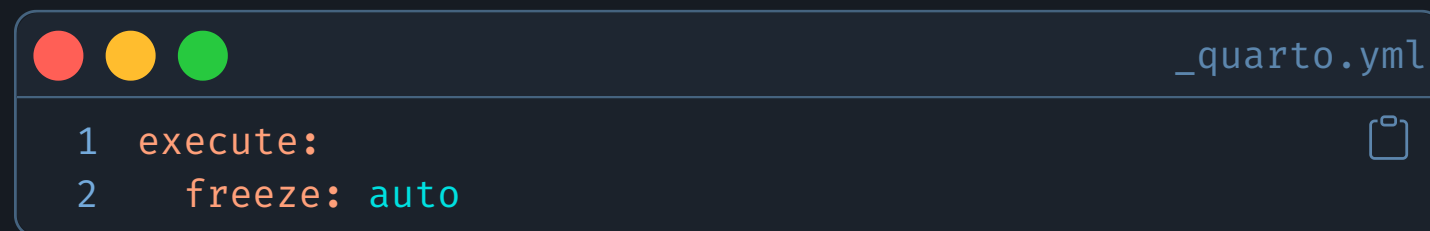
Local Execution + Freeze



```
1 execute:
2   freeze: auto
```

The Big Decision: Where to Execute Code?

Local Execution + Freeze



```
1 execute:
2   freeze: auto
```

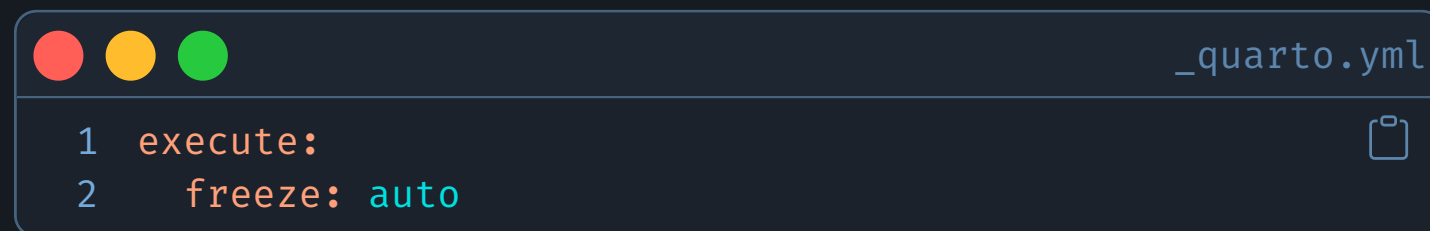
The image shows a terminal window with a dark background. At the top left, there are three colored circles (red, yellow, green). At the top right, the file name `_quarto.yml` is displayed. The main content of the terminal is a YAML configuration snippet with two lines: `1 execute:` and `2 freeze: auto`. A small icon of a document with a checkmark is visible to the right of the second line.

Pros:

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

The Big Decision: Where to Execute Code?

Local Execution + Freeze



```
1 execute:
2   freeze: auto
```

A terminal window with a dark background and light blue text. The title bar shows three colored circles (red, yellow, green) and the filename `_quarto.yml`. The content shows two lines of YAML code: `1 execute:` and `2 freeze: auto`. A small icon of a document with a checkmark is visible on the right side of the terminal window.

Pros:

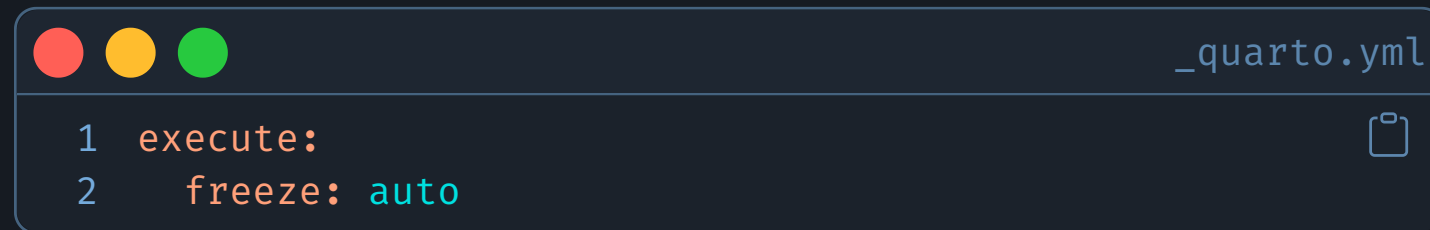
- Faster CI builds.
- Consistent environment.
- Handle legacy code.

Cons:

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

The Big Decision: Where to Execute Code?

Local Execution + Freeze



```
1 execute:
2 freeze: auto
```

Pros:

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

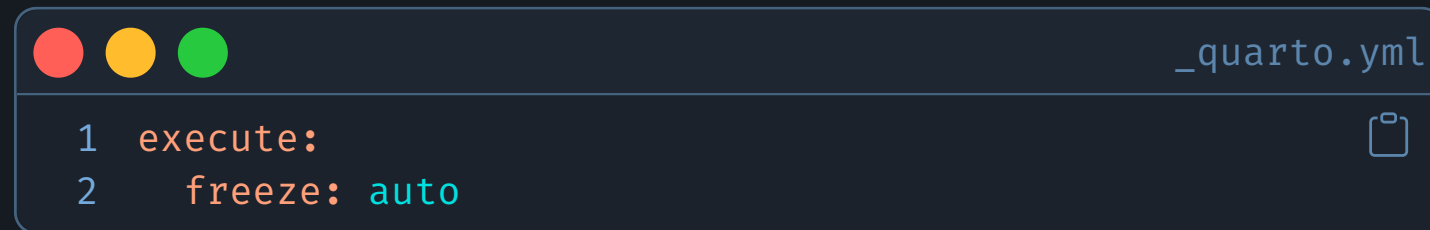
Cons:

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

CI Execution

The Big Decision: Where to Execute Code?

Local Execution + Freeze



```
1 execute:
2   freeze: auto
```

A terminal window with a dark background and light blue text. The title bar shows three colored circles (red, yellow, green) and the filename `_quarto.yml`. The content shows two lines of YAML code: `1 execute:` and `2 freeze: auto`. A small icon of a document with a checkmark is visible in the top right corner of the terminal window.

Pros:

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

Cons:

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

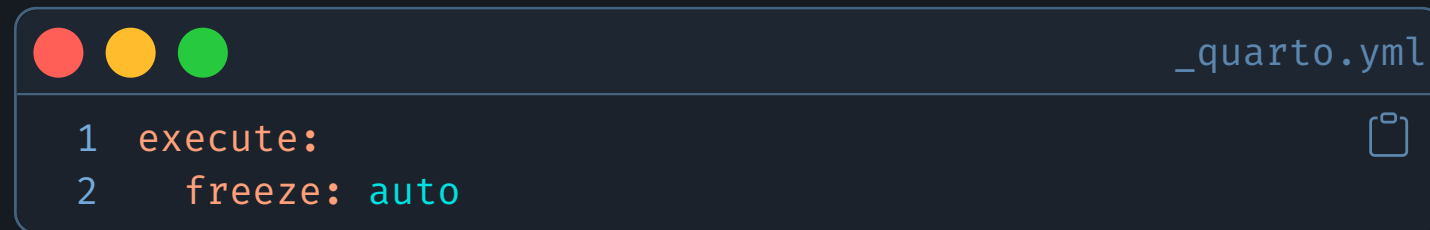
CI Execution

Pros:

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

The Big Decision: Where to Execute Code?

Local Execution + Freeze



```
1 execute:
2   freeze: auto
```

A terminal window with a dark background and light blue text. The title bar shows three colored circles (red, yellow, green) and the filename `_quarto.yml`. The content shows a YAML configuration with two lines: `1 execute:` and `2 freeze: auto`. A small icon of a document with a checkmark is visible in the top right corner of the terminal window.

Pros:

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

Cons:

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

CI Execution

Pros:

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

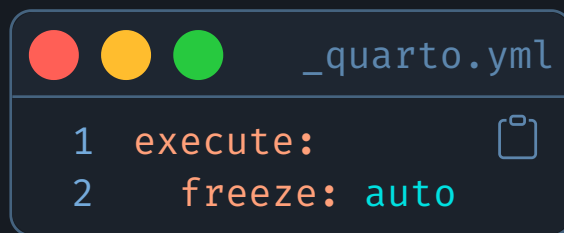
Cons:

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

Freeze Strategy Implementation

Freeze Strategy Implementation

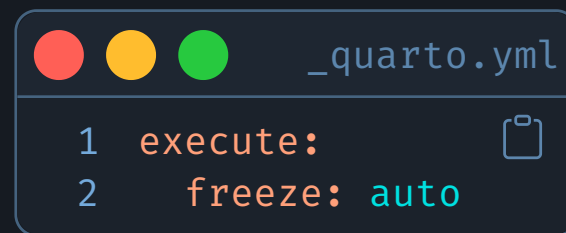
- Enable Freezing



```
1 execute:
2   freeze: auto
```

Freeze Strategy Implementation

- Enable Freezing



```
_quarto.yml
1 execute:
2   freeze: auto
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The title bar reads "_quarto.yml". The terminal content shows a YAML configuration with two lines: "1 execute:" and "2 freeze: auto". A small clipboard icon is visible to the right of the second line.

- Re-render Locally



```
BASH
1 quarto render
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The title bar reads "BASH". The terminal content shows a single line: "1 quarto render". A small clipboard icon is visible to the right of the command.

Freeze Strategy Implementation

- Enable Freezing

```
_quarto.yml  
1 execute:  
2   freeze: auto
```

- Re-render Locally

```
BASH  
1 quarto render
```

- Version Control

```
BASH  
1 git add _freeze/  
2 git commit -m "chore: Update frozen computations"  
3 git push
```


Freeze Strategy Implementation

- Enable Freezing

```
_quarto.yml  
1 execute:  
2   freeze: auto
```

- Re-render Locally

```
BASH  
1 quarto render
```

- Version Control

```
BASH  
1 git add _freeze/  
2 git commit -m "chore: Update frozen computations"  
3 git push
```

! Important

The `_freeze/` directory stores computational results and **must** be committed to Git.

CI Execution with Dependencies

Python/Jupyter Example 

R/Knitr Example 



YAML

```
1 - name: Install Python and Dependencies
2   uses: actions/setup-python@v5
3   with:
4     python-version: '3.13'
5     cache: 'pip'
6
7 - name: Install Jupyter
8   run: pip install jupyter
9
10 - name: Install Requirements
11   run: pip install -r requirements.txt
```

CI Execution with Dependencies

Python/Jupyter Example 

R/Knitr Example 



YAML

```
1 - name: Install R
2   uses: r-lib/actions/setup-r@v2
3   with:
4     r-version: '4.5.0'
5
6 - name: Install R Dependencies
7   uses: r-lib/actions/setup-renv@v2
8   with:
9     cache-version: 1
```



Version Control Strategies

Version Control Strategies

Use a freeze strategy (*recommended*):

YAML

```
1 execute:  
2   freeze: auto
```


Version Control Strategies

Use a freeze strategy (*recommended*):

YAML

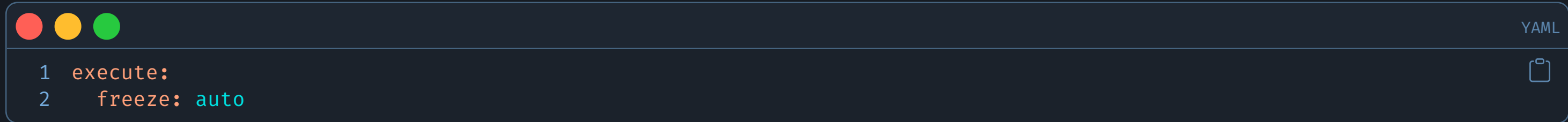
```
1 execute:
2   freeze: auto
```

Commit to Git:

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

Version Control Strategies

Use a freeze strategy (*recommended*):



```
1 execute:
2   freeze: auto
```

The code is displayed in a terminal window with a dark background. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The text 'YAML' is visible in the top-right corner of the window. The code is numbered 1 and 2, with 'execute:' on line 1 and 'freeze: auto' on line 2.

Commit to Git:

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

Don't Commit:

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

Environment Management

Environment Management

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

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.

- Custom Domains

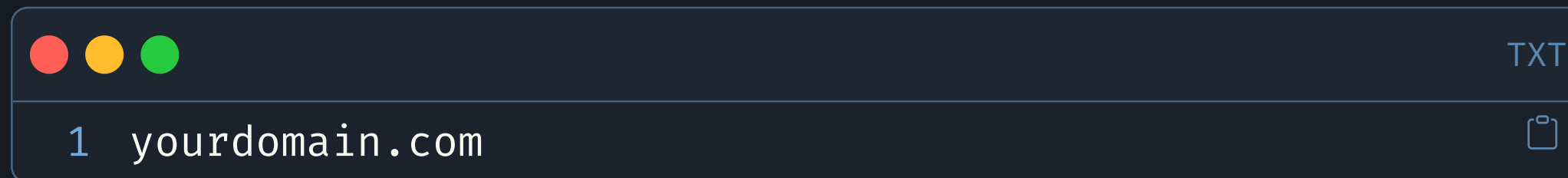
Advanced Topics

Custom Domains

GitHub Pages Custom Domain

Netlify Custom Domain

1. Add **CNAME** file to project root:



```
1 yourdomain.com
```

2. Configure DNS records:

- **Apex domain:** A records to GitHub IPs.
- **Subdomain:** CNAME to **username.github.io**.

3. Enable in repository settings.

Custom Domains

GitHub Pages Custom Domain

Netlify Custom Domain

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

- Distribution Methods
- Publishing Best Practices

Extension Distribution

Distribution Methods

Distribution Methods

GitHub Distribution (Recommended)

Distribution Methods

GitHub Distribution (Recommended)



```
1 # Users install with:
2 quarto add yourorg/company-report
3
4 # Or specific version/tag:
5 quarto add yourorg/company-report@v1.0.0
6
7 # Or from branch:
8 quarto add yourorg/company-report@main
```

Distribution Methods

GitHub Distribution (Recommended)



```
1 # Users install with:
2 quarto add yourorg/company-report
3
4 # Or specific version/tag:
5 quarto add yourorg/company-report@v1.0.0
6
7 # Or from branch:
8 quarto add yourorg/company-report@main
```

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

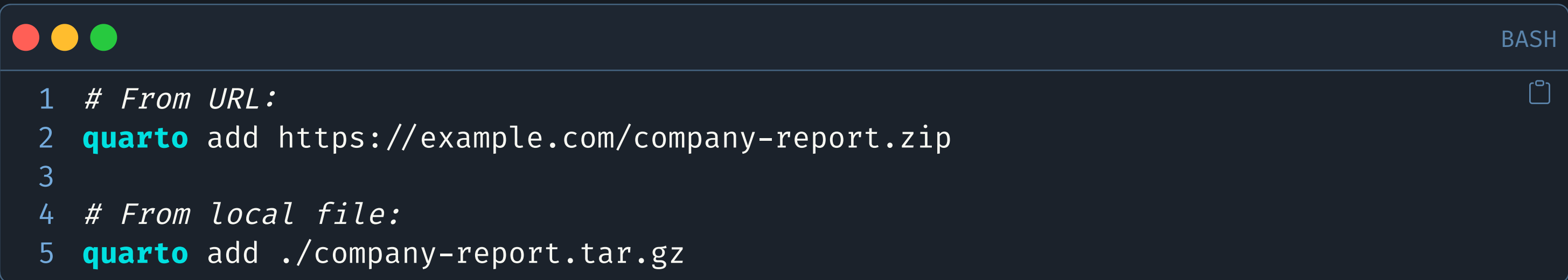
Distribution Methods

Distribution Methods

Archive Distribution

Distribution Methods

Archive Distribution

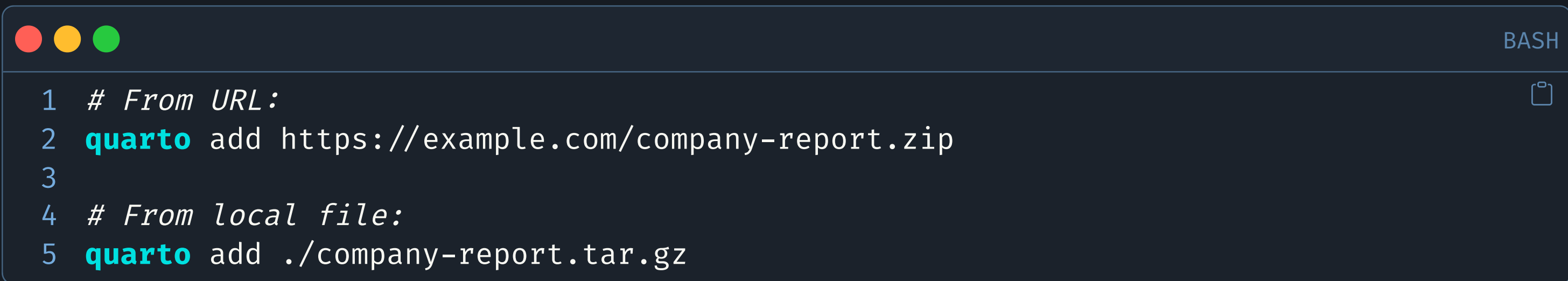


```
1 # From URL:
2 quarto add https://example.com/company-report.zip
3
4 # From local file:
5 quarto add ./company-report.tar.gz
```

BASH

Distribution Methods

Archive Distribution



```
1 # From URL:
2 quarto add https://example.com/company-report.zip
3
4 # From local file:
5 quarto add ./company-report.tar.gz
```

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

Publishing Best Practices

Publishing Best Practices

Prepare your extension:

Publishing Best Practices

Prepare your extension:

1. **Add README.md** - Installation instructions and documentation.

Publishing Best Practices

Prepare your extension:

1. **Add README.md** - Installation instructions and documentation.
2. **Include LICENSE** - Specify usage terms.

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.

Publishing Best Practices

Prepare your extension:

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

Publishing Best Practices

Prepare your extension:

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



Tip

See [Distributing Extensions](#) for complete details.



Mickaël CANOUIL, *Ph.D.*

Independent Contractor

pro@mickael.canouil.dev



Website

mickael.canouil.fr



GitHub

[@mcanouil](https://github.com/mcanouil)



LinkedIn

[MickaelCanouil](https://www.linkedin.com/company/MickaelCanouil)



Bluesky

[@mickael.canouil.fr](https://bsky.app/profile/@mickael.canouil.fr)



Mastodon

[@MickaelCanouil](https://mastodon.social/@MickaelCanouil)