Hosting a Quarto Book Series on Netlify with Route 53 and GitHub Actions

Ronald G. Thomas

2026-06-20

Figure 1: Six bound textbooks whose spines each feed a luminous line into a single domain hub and back out to six separate rendered web pages, symbolizing one domain serving six independently deployed Quarto books.

Introduction

A companion post, Building a statistical computing textbook in the Age of AI, documented the authoring decisions behind the rgtlab Curriculum Project. It ended with a deployment to-do: configure the books to serve from `rgtlab.org`. This post discharges that to-do and generalizes it. The series has since grown from two volumes to six, and all six are now hosted under one domain with one subdomain per book.

The objective is a deployment in which day-to-day editing reduces to `git push`. No manual upload, no DNS edit, and no dashboard interaction should be required to publish a corrected sentence. The configuration that achieves this combines three services:

- **AWS Route 53** as the DNS provider for `rgtlab.org`.

- **Netlify** as the static-site host, one site per book.
- **GitHub Actions** as the continuous-integration system that re-renders each book and re-deploys it on every push to `main`.

The six books, their repositories, and their subdomains are:

Vol	GitHub repo	Subdomain
1	`rgt47/r-bootcamp`	`r-bootcamp.rgtlab.org`
2	`rgt47/practicum`	`practicum.rgtlab.org`
3	`rgt47/scai`	`scai.rgtlab.org`
4	`rgt47/scai-advanced`	`scai-advanced.rgtlab.org`
5	`rgt47/applied-genai`	`applied-genai.rgtlab.org`
6	`rgt47/applied-methods`	`applied-methods.rgtlab.org`

Motivations

- A book under active revision is published continuously, not once. Any step that requires a human to remember a command is a step that eventually gets skipped, leaving the deployed copy stale relative to the source.
- Six independently authored volumes should deploy independently. A render failure in the advanced-methods volume must not block a correction to the boot-camp volume.
- Onboarding a seventh volume should be mechanical: one repository, one Netlify site, one DNS record, with no change to the existing six.
- The deployment should be reproducible from this document alone, so that the configuration survives a lost laptop or a change of maintainer.

Objectives

1. Serve each book at its own subdomain of `rgtlab.org` over HTTPS, with certificates provisioned and renewed automatically.
2. Re-render and re-deploy each book automatically on every push to `main`.
3. Keep the books decoupled so that each deploys on its own schedule and fails in isolation.
4. Document the per-book operation as a loop, so the recipe scales to additional volumes without rewriting.

Figure 2: A cluster of small potted plants of different species, each in an identical terracotta pot: several distinct books served identically from one shared infrastructure.

Prerequisites

This process assumes:

- A registered domain (`rgtlab.org`) with its hosted zone in **AWS Route 53**.
- A **Netlify** account, free tier sufficient.
- A **GitHub** account with one repository per book under a common owner (`rgt47/`).
- Each book is a **Quarto book** (`_quarto.yml` at the repository root) that renders cleanly with `quarto render`.
- **Quarto 1.5+** locally and pinned in CI. The book format changed non-trivially between 1.4 and 1.5.

Hosting options considered

Three configurations were weighed before settling on subdomains.

Subdomains (chosen). Each book gets its own subdomain pointing to a separate Netlify site. DNS is one CNAME record per subdomain. Each book deploys independently. This is the simplest configuration and the one that scales most cleanly: a new volume is one more repository, one more site, and one more record.

Single site with subpaths. One Netlify site rooted at `rgtlab.org` renders all books into subdirectories of a single deploy (`rgtlab.org/docs/scai`). The drawback at six books is that one failed render blocks the entire deploy, coupling volumes that should be independent.

Proxy and rewrites. Each book keeps its own Netlify site, and a 'shell' site for `rgtlab.org` proxies requests via `status = 200` redirects. This preserves decoupling but introduces relative-path edge cases for CSS and image assets, which sometimes resolve at the shell domain rather than the proxied origin and require absolute URLs in each book's Quarto configuration.

The phases below describe the subdomain approach. Where a step says 'for each book', apply it to all six, substituting the relevant repository and subdomain.

Phase 1: Create a Netlify site per book

This phase produces one Netlify site per book and the temporary `*.netlify.app` URLs that DNS will target.

```
npm install -g netlify-cli
netlify login                       # opens browser; authorise
```

From each book's source directory, publish:

```
cd ~/Dropbox/prj/tch/03-scai
quarto publish netlify
```

On first run, Quarto offers to create a new site. Confirm. It builds the book, uploads `_book/`, and prints a URL such as `https://random-name-12345.netlify.app`. Record each URL; Phase 3 needs them.

Phase 2: Add custom subdomains in Netlify

For each site, in the Netlify dashboard:

1. Open **Site configuration** then **Domain management**.
2. Choose **Add a domain** then **Add a domain you already own**.
3. Enter the book's subdomain (for example, `scai.rgtlab.org`), then **Verify** and **Add domain**.
4. Click **Check DNS configuration** and copy the target hostname, which is the site's full `*.netlify.app` name.

Phase 3: Add CNAME records in Route 53

For each subdomain, create one CNAME record in the `rgtlab.org` hosted zone:

- **Record name**: the label only, such as `scai`. Route 53 appends `.rgtlab.org`; do not type the full domain.
- **Record type**: `CNAME`.
- **Value**: that book's `*.netlify.app` hostname, with no `https://` and no trailing slash.
- **TTL**: `300`, so mistakes correct quickly.

- **Routing policy**: Simple routing.

Repeat for `r-bootcamp`, `practicum`, `scai`, `scai-advanced`, `applied-genai`, and `applied-methods`.

Figure 3: A brass mail-sorting rack with several distinct labeled slots, one envelope in each: one subdomain routing to each book.

Phase 4: Verify DNS and SSL

DNS propagation for a fresh subdomain typically takes 1 to 10 minutes. Confirm every record at once:

```
for sub in r-bootcamp practicum scai scai-advanced \
           applied-genai applied-methods; do
  printf '%-32s ' "$sub.rgtlab.org"
  dig "$sub.rgtlab.org" CNAME +short
done
```

Each should return its `*.netlify.app` value. Netlify then requests a Let's Encrypt certificate automatically, which provisions 1 to 5 minutes after DNS resolves. Visit each `https://<subdomain>.rgtlab.org` and confirm it serves over HTTPS with no warnings.

Phase 5: Confirm each book's site-url

Each book's `_quarto.yml` records its public location so that absolute links, social-card metadata, and the sitemap match the deployed URL:

```
cd ~/Dropbox/prj/tch
for d in 01-r-bootcamp 02-practicum 03-scai \
         04-scai-advanced 05-applied-genai 06-applied-methods; do
  printf '%-20s ' "$d"
  grep -h 'site-url' "$d/_quarto.yml"
done
```

Every row should print the matching subdomain. If a URL ever changes, update `site-url`, re-render, and re-publish.

Phase 6: Continuous deployment with GitHub Actions

This phase replaces manual `quarto publish netlify` with automatic deployment on every push to `main`.

First, gather credentials. Generate one Netlify personal access token (**User settings** then **Applications** then **Personal access tokens**), reused across every repository. Then copy each site's **Site ID** (a UUID on its **Site configuration** page).

For each repository, under **Settings** then **Secrets and variables** then **Actions**, add two secrets:

- `NETLIFY_AUTH_TOKEN`: the personal access token.
- `NETLIFY_SITE_ID`: that book's own Site ID.

Then add `.github/workflows/publish.yml`:

```
name: Render and deploy to Netlify

on:
  push:
    branches: [main]
  workflow_dispatch:

jobs:
  publish:
    runs-on: ubuntu-latest
    steps:
      - name: Check out repository
        uses: actions/checkout@v4

      - name: Set up Quarto
        uses: quarto-dev/quarto-actions/setup@v2
        with:
          version: '1.5.57'

      - name: Set up R
        uses: r-lib/actions/setup-r@v2
```

```yaml
        with:
          use-public-rspm: true

      - name: Install R package dependencies
        uses: r-lib/actions/setup-r-dependencies@v2
        with:
          packages: |
            any::knitr
            any::rmarkdown
            any::dplyr
            any::ggplot2
            any::survival
            any::palmerpenguins

      - name: Render book
        run: quarto render

      - name: Deploy to Netlify
        uses: nwtgck/actions-netlify@v3
        with:
          publish-dir: ./_book
          production-branch: main
          production-deploy: true
          deploy-message: 'GHA: ${{ github.event.head_commit.message }}'
          enable-pull-request-comment: false
          overwrites-pull-request-comment: false
        env:
          NETLIFY_AUTH_TOKEN: ${{ secrets.NETLIFY_AUTH_TOKEN }}
          NETLIFY_SITE_ID: ${{ secrets.NETLIFY_SITE_ID }}
        timeout-minutes: 5
```

The `packages:` list is per-book. Each workflow declares only the R packages that book evaluates during render, so the lists differ across repositories. The methods and advanced-methods volumes pull in `lme4`, `Matrix`, and `survival`; the boot-camp volume needs fewer. When a chapter introduces a new dependency, add it to that book's list; a CI failure will name what is missing.

Verification

After the first push, open the repository's **Actions** tab and watch the run. The first run takes 4 to 8 minutes, dominated by R package installation; later runs take 2 to 3 minutes once the package cache is warm. A green run means the deployed subdomain is live with the new content.

Things to Watch Out For

1. **CNAME value must be bare.** Route 53 rejects, or silently mis-resolves, a value that includes `https://` or a trailing slash. Enter only the `*.netlify.app` hostname.
2. **SSL lags DNS.** The Let's Encrypt certificate provisions only after DNS resolves. A site that loads over HTTP but warns on HTTPS is usually mid-provisioning, not misconfigured. Wait a few minutes before debugging.
3. **Each repository needs its own Site ID.** The auth token is shared, but `NETLIFY_SITE_ID` is per book. Pasting one book's Site ID into another's secrets deploys the wrong book to the wrong subdomain.
4. **Per-book package lists drift.** A chapter ported from one volume into another may carry a dependency the destination book's workflow does not install. The render fails in CI, not locally, because the local machine already has the package.
5. **Pin the Quarto version in CI.** The book format changed between 1.4 and 1.5. An unpinned `setup` action that floats to a future major version can change rendering behavior without any source edit.

Optional: the freeze cache

Books with `execute: freeze: auto` can commit the `_freeze/` directory so CI reuses cached chunk output instead of re-executing every chunk on every push:

```
echo '!_freeze/' >> .gitignore        # un-ignore if needed
git add _freeze
git commit -m "Commit Quarto freeze cache"
git push
```

CI then re-executes only chunks whose source changed. The trade-off is repository growth, acceptable for the small, deterministic chunks typical of these books.

Figure 4: A wax letter seal being pressed into fresh red wax on an envelope: automatic SSL certification sealing every deploy.

What Did We Learn?

- **Subdomains decouple deployment.** One site per book means a failed render in one volume never blocks another, and onboarding a new volume touches nothing in the existing ones.
- **`git push` is the whole workflow.** Once secrets and the workflow file are in place, publishing a correction requires no command beyond the commit that makes it.
- **Independence is configuration, not discipline.** The decoupling holds because each repository carries its own Site ID, its own package list, and its own workflow, not because the author remembers to keep them separate.
- **The recipe scales by repetition.** Every phase is a per-book loop. A seventh volume is the same six steps once more, plus one DNS record.

See Also

- Building a statistical computing textbook in the Age of AI: the authoring decisions behind the books deployed here.

Key resources:

- Quarto: Publishing to Netlify
- Netlify custom domains
- AWS Route 53 documentation

- nwtgck/actions-netlify

Reproducibility

Component	Version
Quarto	1.5.57
Netlify CLI	current
GitHub Actions runner	ubuntu-latest
Last verified	2026-06-20

Draft. Hero image in place; `draft: true` *pending a* `quarto render` *check and wiring into the post compendium.*

Related posts in this cluster

This post is part of the *Quarto, R Markdown, and Publishing* series. Recommended reading order:

1. Multi-Language Quarto Documents on macOS
2. Rapid Conversion of Draft R Scripts to Formal Rmd
3. Building a Statistical Computing Textbook
4. Producing R Coding Videos: OBS Screencasts and Narrated Slide Decks
5. **Hosting a Quarto Book Series on Netlify** (this post)
6. A Review and Proposal for Policy Regarding Generative AI Use in Graduate Biostatistics Courses