From Code to Content: Automating Your Blogger Workflow with GitHub

Frank Jung

2026-02-07

Contents

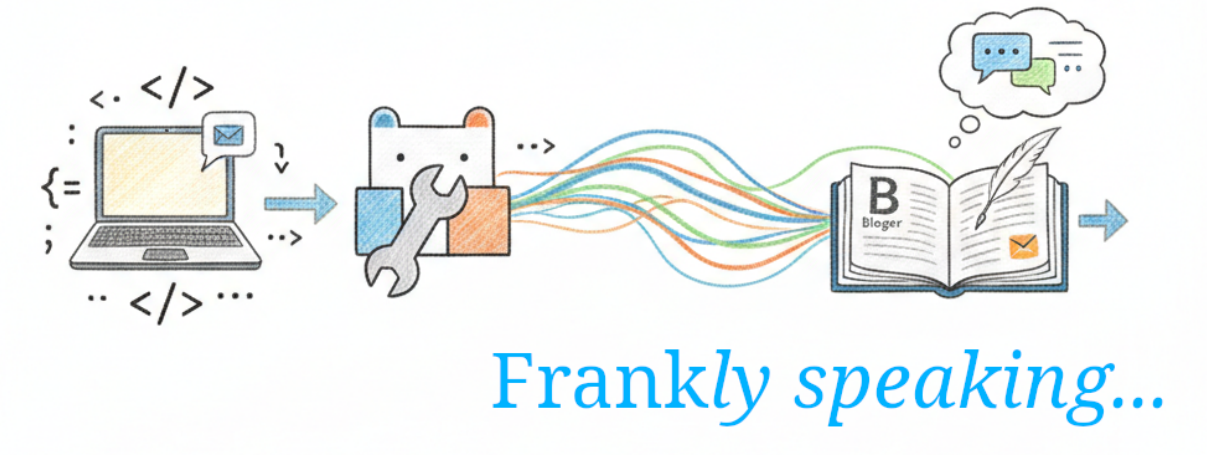

Figure 1: © Frank H Jung 2026

For the modern developer, the desire to share technical insights is often hindered by the "toil" of the publishing process. We live in our code editors and terminal sessions, yet sharing those insights on platforms like Blogger traditionally requires a regression to manual labour: copy-pasting HTML, wrestling with web-based WYSIWYG editors, and manually managing assets. This discourages regular contribution, often resulting in stale repositories and abandoned drafts.

By adopting a "Blogging as Code" approach, you can replace manual copy-pasting with automated CI/CD pipeline management. By leveraging the Blogger REST API v3 and GitHub Actions, you can treat your blog with the same engineering rigour as your production software.

1. The API as Your Deployment Interface

The core shift in this methodology is viewing Blogger not as a website, but as a deployment target. The Blogger REST API allows developers to bypass the browser entirely, enabling programmatic creation and updates of posts.

Treating your blog as a deployment target via an API offers three critical advantages for the DevOps-minded writer:

- Version Control: Your content resides in a Git repository, providing a single source of truth and a complete audit trail of changes.
- Consistency: Automation ensures that metadata, labels, and formatting are applied uniformly, eliminating inconsistent posts.
- GitOps for Content: Merging a pull request to your main branch becomes the trigger for your live content updates.

2. Security Starts in the Google Cloud Console

Automation requires a bridge of trust between GitHub and Google. This is established in the Google Cloud Console by enabling OAuth 2.0. Setting up OAuth 2.0 involves a one-time configuration investment that yields long-term efficiency gains.

For a detailed guide on setting up OAuth 2.0 and generating your refresh token see Google Blogger API Authentication Setup.

You must explicitly search for and enable the Blogger API within your project.

Once the API is active, you will generate two critical credentials:

1. CLIENT_ID
2. CLIENT_SECRET

These are not just identifiers; they are the foundation used to generate a REFRESH_TOKEN. Because GitHub Actions run in a non-interactive environment, this refresh token is the key to secure, long-term authentication, allowing the workflow to request fresh access tokens without manual intervention or the storage of your primary account password.

3. Knowing Your Unique Identifiers

To route your content correctly, the pipeline must identify your specific destination. This is handled by your BLOG_ID. Together, these parameters form the mandatory authentication and identification components required for your pipeline.

Parameter	Description
BLOG_ID	The unique identifier for your specific destination blog.
CLIENT_ID	The Google OAuth Client ID derived from the Cloud Console.
CLIENT_SECRET	The Google OAuth Client Secret used for authorization.
REFRESH_TOKEN	The secure token that allows for long-term, non-interactive API access.

[!WARNING] Security best practice: It is essential that these four values are stored as GitHub Secrets. Hardcoding these credentials in your YAML or repository exposes your Google account to the public. By using GitHub Secrets, they are encrypted and injected into the runner environment only during execution.

4. The Anatomy of an Automated Post

The heavy lifting of the API interaction is managed by the frankhjung/blogger@v1 GitHub Action. This utility expects a specific set of inputs to govern how your content is delivered:

- source-file: The path to the rendered HTML content.
- title: The headline of your post.
- labels: A comma-separated list of tags (e.g., “tech, devops, tutorial”) to organise your content.
- blog-id: Your destination identifier.
- client-id / client-secret / refresh-token: Your secure authentication suite.

The action also provides several advanced features to streamline the process:

- Embedded Assets: Local images referenced in your HTML are automatically encoded as Base64 data URIs, ensuring they are correctly displayed without requiring separate hosting.
- Smart Extraction: If a full HTML document (with <html> and <body> tags) is provided, the action intelligently extracts only the body content and internal CSS styles, ensuring it integrates perfectly with your blog’s template.

The source-file input expects HTML. While the Blogger API consumes HTML, your source remains in the format you prefer—such as Markdown or R Markdown. This is handled by a “build” stage in your workflow that converts your source code into the final deployment artifact.

5. Updates Without the Duplication Headache

A key feature of this automated approach is idempotency. In a traditional manual workflow, fixing a typo means hunting through a dashboard. In an automated system without idempotency, re-running a pipeline would simply create a duplicate post.

The blogger utility avoids this by using the post title as a unique identifier. If a draft post with the same title already exists, it updates the existing content and labels. Note that live or scheduled posts are not

modified to prevent accidental overwrites. Include the title as an environment variable in your GitHub pipeline. If you change the title, then a new post will be created.

This supports a truly iterative writing process: push to main, and the Action will find the existing draft post and update its content.

Note:

Because the title is the identifier, changing the title in your source file will result in a new post being created. Additionally, to provide a final safety check, all new posts are created as drafts by default, giving one last opportunity for a manual preview before going live.

6. Bringing It All Together in YAML

A Concrete Example: R Markdown with Data Visualisations

To see this in practice, consider the article-base-rate repository. This uses complex data visualisations. The content is written in R Markdown (.Rmd), requiring R (version 4.0+) and a Makefile to render the final HTML.

The .github/workflows/publish.yml integrates the build and deployment stages into a single, cohesive CI/CD pipeline:

```yaml
name: Publish to Blogger
on:
  push:
    branches: [ main ]

jobs:
  build:
    name: publish article to pages
    runs-on: ubuntu-latest
    env:
      ARTICLE_TITLE: "your title here"
      LABELS: "label_1, label_2"
    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      # ... tasks to render markdown to HTML ...

      - name: publish to blog
        if: success()
        uses: docker://frankhjung/blogger:v1
        with:
          args: >-
```



```
--source-file "path/to/your/article.html"
--title "Your Title Here"
--labels "label_1, label_2"
--blog-id "${{ secrets.BLOGGER_BLOG_ID }}"
--client-id "${{ secrets.BLOGGER_CLIENT_ID }}"
--client-secret "${{ secrets.BLOGGER_CLIENT_SECRET }}"
--refresh-token "${{ secrets.BLOGGER_REFRESH_TOKEN }}"
```

This configuration mirrors the logic of deploying to GitHub Pages but redirects the final, styled output to the Blogger platform, bridging the gap between sophisticated data analysis and public outreach.

7. Handling Format Constraints

The Blogger API expects post content as HTML, even if you prefer writing in Markdown (as you would for a wiki or documentation site). The compromise is a simple build step that converts your source into a single HTML file.

In practice, that usually looks like one of these:

- Markdown → HTML via Pandoc.
- R Markdown → HTML via rmarkdown.

The output HTML becomes the workflow artefact you pass as source-file (for example, public/index.html).

Conclusion: The Future of Your Technical Blog

Treating Blogger as a deployment target lets you apply the same engineering discipline to writing that you already apply to software: version-controlled content, repeatable builds, and automated publishing.

Once the pipeline is in place, publishing becomes routine: write in the format you like, render to HTML, and let GitHub Actions update the existing post via the API.

More Information

- Blogger REST API v3
- Blogger Publishing Action:
- Example repository: article-base-rate
- Example repository: article-publish-to-blogspot
- GitHub Actions for Blogspot
- Git
- GitHub Actions
- GitHub
- Markdown guide
- Pandoc
- R Markdown

