From Code to Content: Automating Your Blogger Workflow with GitHub

Frank Jung

2026-02-03

Contents

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 (Blogspot) traditionally requires a regression to manual labour: copy-pasting HTML, wrestling with web-based WYSIWYG editors, and manually managing assets. This cognitive load creates a significant barrier to entry, often resulting in stale repositories and abandoned drafts.

It is time to eliminate this friction. By adopting a "Blogging as Code" manifesto, you can transform your workflow into a seamless, automated CI/CD pipeline. By leveraging the Blogger REST API v3 and GitHub Actions, you can treat your blog with the same engineering rigor as your production software.

1. The API is Your Most Powerful Ally

The core shift in this methodology is viewing Blogger not as a website, but as a deployment target. The Blogger REST API v3 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 “snowflake” 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. To begin, 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 “Big Four” authentication and identification components required for your pipeline.

Parameter	Description
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.
BLOG_ID	The unique identifier for your specific destination blog.

[!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/blogspot-publishing@main GitHub Action. This utility expects a specific set of inputs to govern how your content is delivered:

- title: The headline of your post.

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

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

One of the most elegant features of this automated approach is Idempotency. In a traditional manual workflow, fixing a typo means hunting through a dashboard. In a naive automated system, re-running a pipeline would simply create a duplicate post.

The blogspot-publishing utility avoids this by using the post title as a unique identifier. As the source material confirms:

“If a post with the same title already exists, it updates the existing post (content only) instead of creating a duplicate.”

This supports a truly iterative writing process. Correct a technical error in your repository, push to main, and the Action will find the existing 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 you one last opportunity for a manual preview before going live.

6. Bringing It All Together in YAML

To see this in practice, consider a repository like article-base-rate, which 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.

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

```
---

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

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
```



```yaml
- name: Checkout code
  uses: actions/checkout@v4

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

- name: Install dependencies and render
  run: |
    Rscript -e 'install.packages(c("rmarkdown", "ggplot2", "knitr"))'
    make base-rate.html

- name: Publish to Blogspot
  uses: frankhjung/blogspot-publishing@main
  with:
    title: "Base Rate Fallacy Explained"
    source-file: "public/index.html"
    blog-id: ${{ secrets.BLOGGER_BLOG_ID }}
    client-id: ${{ secrets.BLOGGER_CLIENT_ID }}
    client-secret: ${{ secrets.BLOGGER_CLIENT_SECRET }}
    refresh-token: ${{ secrets.BLOGGER_REFRESH_TOKEN }}
    labels: "statistics, r-programming, data-science"
```

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.

Conclusion: The Future of Your Technical Blog

By migrating your workflow from manual uploads to a sophisticated CI/CD pipeline, you treat your intellectual output with the same respect as your codebase. Automation removes the friction of publishing, ensuring that your insights move from your editor to your audience with maximum security and minimal toil.

Now that your publishing is automated, what complex technical story will you tell next?

References

- Blogger REST API v3
- Git
- GitHub Actions
- GitHub
- Example: GitHub Actions for Blogger
- Example: article-base-rate Repository

