

Hylograph Cloudflare Deployment Plan

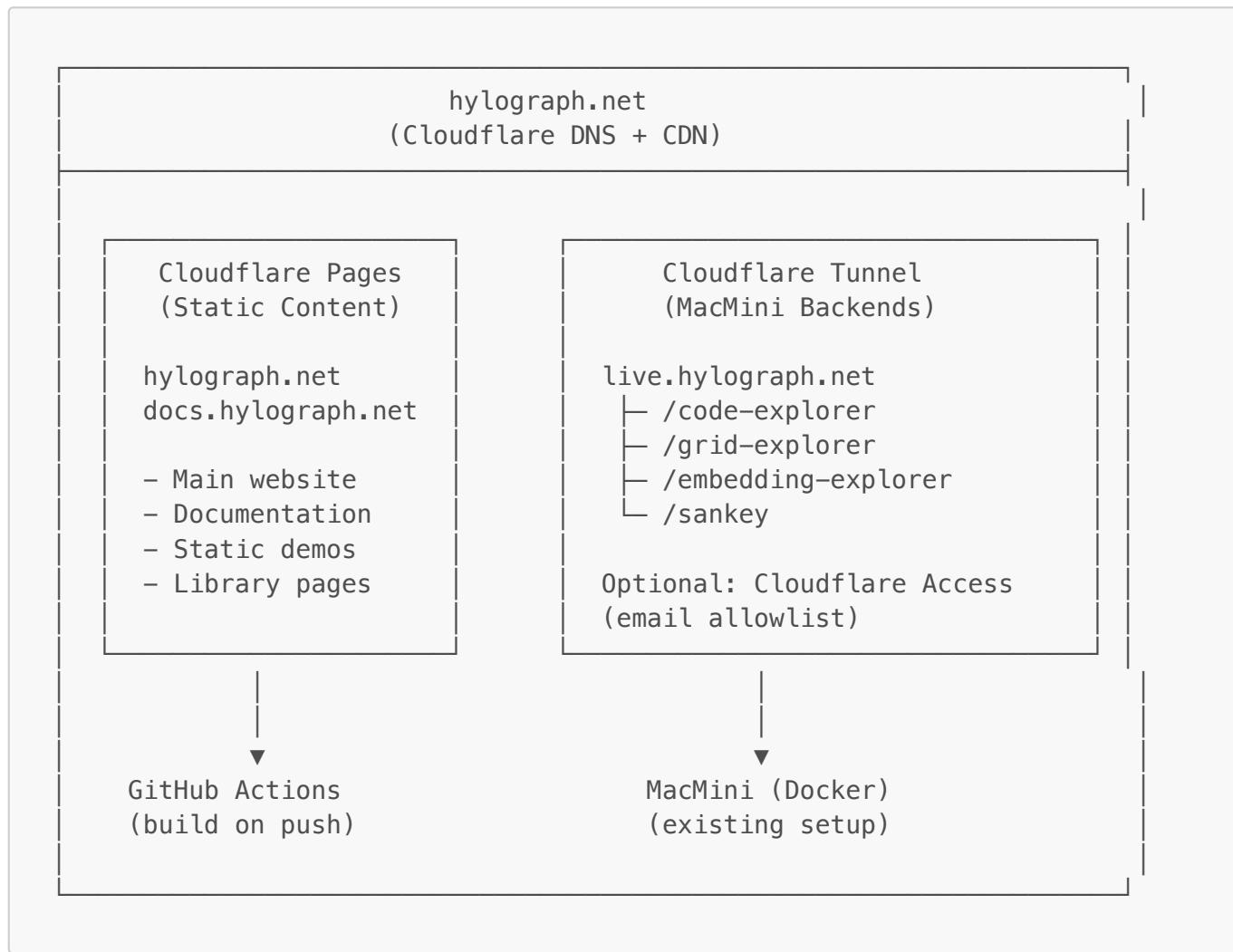
Overview

Deploy the Hylograph project to `hylograph.net` using Cloudflare's free tier:

- **Static content** → Cloudflare Pages
- **Backend demos** → Cloudflare Tunnel to MacMini

Initial audience: friends and PureScript Discord community. All repos will be public.

Architecture



Domain Structure

URL	Content	Hosting	Notes
<code>hylograph.net</code>	Main site + static demos	Cloudflare Pages	Rebrand from PSD3
<code>docs.hylograph.net</code>	Antora documentation	Cloudflare Pages	Separate Pages project
<code>live.hylograph.net</code>	Backend demos	Tunnel → MacMini	Access-controlled

Alternative (simpler, single Pages project):

URL	Content
hylograph.net/	Main site
hylograph.net/docs/	Documentation
hylograph.net/demos/	Static demos
live.hylograph.net/	Backend demos

Recommendation: Start with the simpler single-project approach. Split later if needed.

Content Inventory

Tier 1: Static (Cloudflare Pages)

Current Name	New Path	Source	Build
Main website	/	site/website	make website
Documentation	/docs/	site/docs	npx antora
WASM Demo	/demos/wasm/	wasm-force-demo	make app-wasm
AST Builder	/demos/ast-builder/	TBD after HATS	make app-ast-builder
Sankey (static)	/demos/sankey/	psd3-arid-keystone	make app-sankey (static mode)
Library: Selection	/libs/selection/	site/lib-selection	make lib-site-selection
Library: Simulation	/libs/simulation/	site/lib-simulation	make lib-site-simulation
Library: Layout	/libs/layout/	site/lib-layout	make lib-site-layout
Library: Graph	/libs/graph/	site/lib-graph	make lib-site-graph
Library: Music	/libs/music/	site/lib-music	make lib-site-music

Tier 2: Backend Required (Cloudflare Tunnel)

Demo	Backend	Static Fallback
Code Explorer	Node.js (ce-server)	Link to repo + build instructions
Grid Explorer	Python (PurePy)	Link to repo + build instructions
Embedding Explorer	Python (PurePy)	Link to repo + build instructions
Sankey Editor	Optional backend	Static file mode with sample data
Tidal Editor	Erlang (Purerl)	Complex - may skip initially

Implementation Steps

Phase 1: Cloudflare Setup

1.1 DNS Configuration (Cloudflare Dashboard)

Since domain is registered with Cloudflare, DNS is already managed there.

Add records:

Type	Name	Content	Proxy
A	@	192.0.2.1	Yes (placeholder for Pages)
CNAME	docs	<pages-project>.pages.dev	Yes
CNAME	live	<tunnel-id>.cfargotunnel.com	Yes

Note: Cloudflare Pages automatically configures DNS when you add a custom domain.

1.2 Create Cloudflare Pages Project

1. Go to Cloudflare Dashboard → Pages → Create a project
2. Connect to GitHub repository (will need to be public or grant access)
3. Configure build settings:

```
Build command: make cloudflare-build
Build output directory: dist/cloudflare
Root directory: /
```

4. Add custom domain: [hylograph.net](#)

1.3 Add Makefile Target

```
# New target for Cloudflare Pages build
cloudflare-build:
    @echo "Building for Cloudflare Pages..."
    mkdir -p dist/cloudflare

    # Main website
    $(MAKE) website
    cp -r site/website/public/* dist/cloudflare/

    # Documentation (if Antora is available)
    cd site/docs && npx antora antora-playbook.yml || true
    mkdir -p dist/cloudflare/docs
    cp -r site/docs/build/site/* dist/cloudflare/docs/ || true

    # Static demos
    $(MAKE) app-wasm
```

```

mkdir -p dist/cloudflare/demos/wasm
cp -r showcases/wasm-force-demo/public/* dist/cloudflare/demos/wasm/

# Library sites
$(MAKE) lib-sites
mkdir -p dist/cloudflare/libs
for lib in selection simulation layout graph music; do \
    mkdir -p dist/cloudflare/libs/$$lib; \
    cp -r site/lib-$$lib/public/* dist/cloudflare/libs/$$lib/; \
done

@echo "Build complete: dist/cloudflare/"

```

1.4 Create _redirects File

Create `dist/cloudflare/_redirects` for SPA routing and backend proxying:

```

# SPA fallback for main site
/*      /index.html      200

# Redirect old PSD3 URLs (if any)
/psd3/*      /:splat      301

```

Or use `_headers` for caching:

```

/*
  Cache-Control: public, max-age=3600

/assets/*
  Cache-Control: public, max-age=31536000, immutable

```

Phase 2: Cloudflare Tunnel Setup (MacMini)

2.1 Install cloudflared on MacMini

```

# On MacMini
brew install cloudflared

```

2.2 Authenticate and Create Tunnel

```

# Login (opens browser)
cloudflared tunnel login

# Create tunnel

```

```
cloudfaired tunnel create hylograph-macmini  
# Note the tunnel ID and credentials file path
```

2.3 Configure Tunnel

Create `~/.cloudfaired/config.yml` on MacMini:

```
tunnel: <TUNNEL_ID>  
credentials-file: /Users/andrew/.cloudfaired/<TUNNEL_ID>.json  
  
ingress:  
  # Code Explorer  
  - hostname: live.hylograph.net  
    path: /code-explorer/*  
    service: http://localhost:8085  
  
  # Code Explorer API  
  - hostname: live.hylograph.net  
    path: /code/*  
    service: http://localhost:3000  
  
  # Grid Explorer  
  - hostname: live.hylograph.net  
    path: /grid-explorer/*  
    service: http://localhost:8088  
  
  # Embedding Explorer  
  - hostname: live.hylograph.net  
    path: /embedding-explorer/*  
    service: http://localhost:8087  
  
  # Sankey Editor (with backend)  
  - hostname: live.hylograph.net  
    path: /sankey/*  
    service: http://localhost:8089  
  
  # Catch-all (404)  
  - service: http_status:404
```

2.4 Add DNS Route

```
cloudfaired tunnel route dns hylograph-macmini live.hylograph.net
```

2.5 Run Tunnel as Service

```
# Install as launchd service (macOS)
sudo cloudflared service install

# Or run manually for testing
cloudflared tunnel run hylograph-macmini
```

Phase 3: Access Control (Optional)

If you want to limit access to live.hylograph.net:

3.1 Cloudflare Access Setup

1. Cloudflare Dashboard → Zero Trust → Access → Applications
2. Add application:
 - Name: Hylograph Live Demos
 - Domain: live.hylograph.net
 - Session duration: 24 hours
3. Add policy:
 - Name: Discord Community
 - Action: Allow
 - Include: Emails ending in specific domains OR specific email addresses

Free tier includes 50 users, which is plenty for initial community testing.

3.2 Alternative: No Access Control

For fully public access, skip this step. Cloudflare's default DDoS protection and rate limiting will provide basic protection.

Phase 4: CI/CD (GitHub Actions)

Create `.github/workflows/deploy-pages.yml`:

```
name: Deploy to Cloudflare Pages

on:
  push:
    branches: [main]
  workflow_dispatch:

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
```

```

node-version: '20'

- name: Setup PureScript
  uses: purescript-contrib/setup-purescript@main
  with:
    purescript: '0.15.15'
    spago: '0.21.0'

- name: Install dependencies
  run: npm ci

- name: Build for Cloudflare
  run: make cloudflare-build

- name: Deploy to Cloudflare Pages
  uses: cloudflare/pages-action@v1
  with:
    apiToken: ${{ secrets.CLOUDFLARE_API_TOKEN }}
    accountId: ${{ secrets.CLOUDFLARE_ACCOUNT_ID }}
    projectName: hylograph
    directory: dist/cloudflare

```

Required secrets:

- **CLOUDFLARE_API_TOKEN**: Create in Cloudflare Dashboard → API Tokens
- **CLOUDFLARE_ACCOUNT_ID**: Found in Cloudflare Dashboard URL or overview page

Phase 5: Rebranding

5.1 Update Main Website

- Change branding from PSD3 to Hylograph
- Update navigation to use new URL structure
- Add "Live Demos" link to live.hylograph.net (if access-controlled, note this)
- Add prominent "View Source" / GitHub links

5.2 Update Documentation

- Rebrand Antora site
- Update installation instructions with new package names (if changing)
- Add "Try it live" links where applicable

5.3 Static Demo Fallbacks

For demos that need backends, create static fallback pages:

```

<!-- demos/code-explorer/index.html (static fallback) -->
<h1>Code Explorer</h1>
<p>This demo requires a backend server.</p>
<h2>Try it live</h2>

```

```
<p><a href="https://live.hylograph.net/code-explorer/">live.hylograph.net/code-explorer</a></p>
<h2>Run locally</h2>
<pre>
git clone https://github.com/youruser/corrode-expel
cd corrode-expel
make build
make run
# Open http://localhost:8085
</pre>
```

Cost Analysis

Cloudflare Free Tier includes:

- Pages: Unlimited sites, 500 builds/month, 100GB bandwidth
- Tunnel: Unlimited, free
- Access: 50 users
- DDoS protection: Included
- Rate limiting: 10,000 requests/day (free rule)

Estimated monthly cost: \$0

For higher traffic or more Access users, paid plans start at \$5/month (Pro).

Migration Checklist

- Verify hylograph.net DNS is active in Cloudflare
- Create Cloudflare Pages project
- Add `cloudflare-build` Makefile target
- Test build locally: `make cloudflare-build`
- Connect Pages to GitHub repo
- Add custom domain to Pages project
- Verify static site deploys correctly
- Install cloudflared on MacMini
- Create and configure tunnel
- Verify existing Docker services still work
- Configure tunnel ingress rules
- Test tunnel connectivity
- (Optional) Set up Cloudflare Access
- Update website branding
- Announce to community

Rollback Plan

If issues arise:

1. Static content: Revert GitHub commit, Pages auto-redeploys
2. Tunnel: Stop cloudflared service, backends become unreachable (fail closed)

3. DNS: Point hylograph.net back to MacMini directly (remove Pages)

The MacMini deployment remains unchanged and can serve as fallback.

Future Considerations

- **Custom domain for docs:** docs.hylograph.net as separate Pages project if docs grow large
 - **Preview deployments:** Cloudflare Pages creates preview URLs for PRs automatically
 - **Analytics:** Cloudflare Web Analytics (free, privacy-respecting)
 - **Caching:** Fine-tune cache headers for assets vs HTML
 - **Monitoring:** Cloudflare Dashboard shows traffic, errors, performance
-

Status / Next Steps

1. Confirm DNS is active for hylograph.net
2. Create `cloudflare-build` Makefile target
3. Test local build
4. Create Cloudflare Pages project
5. Set up GitHub Actions workflow
6. Configure Cloudflare Tunnel on MacMini
7. Test end-to-end
8. Announce to PureScript Discord