Deployment Appendix — Security Headers (Catalog Template)

Overview

- These snippets mirror the app-layer headers set in `frontend/web/next.config.js` and should be applied at your edge or origin so intermediate proxies/CDNs do not remove them.
- Replace `example.com`, `cdn.example.com`, and other placeholder hosts with your real origins and trusted third-party services.
- **Important:** Only enable HSTS on HTTPS listeners in production.

Nginx (server block)

```nginx
server {
    listen 443 ssl;
    server_name example.com;
    root /var/www/html;
    add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload" always;
    add_header X-Frame-Options "DENY" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "no-referrer-when-downgrade" always;
    add_header Cross-Origin-Opener-Policy "same-origin" always;
    add_header Content-Security-Policy "default-src 'self'; script-src 'self' https://cdn.example.com; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com; img-src 'self' data: https://images.example.com; font-src 'self' https://fonts.gstatic.com; connect-src 'self' https://api.example.com; frame-ancestors 'none';" always;
    # Serve site...
}
```

Apache (VirtualHost)

```apache
<VirtualHost *:443>
    ServerName example.com
    DocumentRoot /var/www/html
    Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload"
    Header always set X-Frame-Options "DENY"
    Header always set X-Content-Type-Options "nosniff"
    Header always set Referrer-Policy "no-referrer-when-downgrade"
    Header always set Cross-Origin-Opener-Policy "same-origin"
    Header always set Content-Security-Policy "default-src 'self'; script-src 'self' https://cdn.example.com; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com; img-src 'self' data: https://images.example.com; frame-ancestors 'none';"
</VirtualHost>
```

CloudFront (Response Headers Policy)

- Use the `scripts/generate_cloudfront_policy.ps1` helper to produce a `response-headers-policy.json` file and then apply it with the AWS CLI.

Quick validation commands

```bash
curl -I https://example.com/ | egrep -i 'strict-transport-security|content-security-policy|x-frame-options|referrer-policy|cross-origin-opener-policy'
```

Operational notes

- Ensure the CSP string exactly matches the runtime needs of your application (e.g., `connect-src` for API endpoints, `img-src` for image hosts, `script-src` for trusted CDNs).
- Test in staging first, then enable HSTS in production only when TLS termination is confirmed and certificates are managed.
- If you use a reverse proxy or additional CDN in front of CloudFront, ensure headers are not overwritten; prefer applying at the edge closest to clients.

Optional follow-ups

- Provide a CloudFormation/SAM snippet to create and attach the CloudFront Response Headers Policy — say if you want I can generate a CFN template next.
- Generate a multi-size `.ico` from the branding assets (not required for catalog readiness).

Attach / Update CloudFront Distribution (manual steps)

1) Deploy the CloudFormation stack (or create the policy via the AWS CLI):

```bash
aws cloudformation deploy --template-file infrastructure/cloudfront-response-headers.yaml --stack-name catalog-response-headers
```

2) Retrieve the created policy ID from the stack outputs:

```bash
aws cloudformation describe-stacks --stack-name catalog-response-headers --query 'Stacks[0].Outputs'
```

3) Attach the Response Headers Policy to your distribution (example flow):

```bash
# Fetch the distribution config and ETag
aws cloudfront get-distribution-config --id <DISTRIBUTION_ID> > dist-config.json
ETAG=$(aws cloudfront get-distribution-config --id <DISTRIBUTION_ID> --query 'ETag' --output text)
# Edit dist-config.json: set your ResponseHeadersPolicyId under
# DistributionConfig.DefaultCacheBehavior.ResponseHeadersPolicyId or per-CacheBehavior
# Update the distribution
aws cloudfront update-distribution --id <DISTRIBUTION_ID> --distribution-config file://dist-config.json --if-match $ETAG
```

Notes:

- You can also attach the policy via the CloudFront console by editing the Behavior and selecting the Response Headers Policy.
- If you have multiple Behaviors, attach the policy to each relevant behavior.

Cloudflare (HTTP Response Headers Rules)

- Prefer Cloudflare's Response Headers Rules to mirror app-layer headers; no Workers required.
- Create a rule with condition `URL matches *` and set the following response headers (replace hosts as needed):

Content-Security-Policy

```
default-src 'self';
script-src 'self';
style-src 'self' 'unsafe-inline';
img-src 'self' data:;
font-src 'self';
connect-src 'self';
frame-ancestors 'none';
base-uri 'self';
form-action 'self';
```

Strict-Transport-Security

```
max-age=63072000; includeSubDomains; preload
```

Cross-Origin-Opener-Policy

```
same-origin
```

X-Frame-Options

```
```

```
DENY
```

Referrer-Policy

```
no-referrer-when-downgrade
```

Order of operations (Cloudflare)

1. Next.js application-layer headers (already set in `frontend/web/next.config.js`)
2. Cloudflare Response Headers Rule (apply with `URL matches *`)
3. Verify that Cloudflare does not unintentionally override or duplicate application headers

Verification

Run a quick check after deploy to confirm headers are present and correct:

```bash
curl -I https://example.com/ | egrep -i 'strict-transport-security|content-security-policy|x-frame-options|referrer-policy|cross-origin-opener-policy'
```

Open the browser DevTools Security / Network panel and verify there are no duplicate or conflicting header values.

Change control note

Treat this appendix as the authoritative reference for deployment-time header configuration. If you change headers in `frontend/web/next.config.js`, update this appendix and notify ops/security so the CDN/origin rules can be kept in sync.

Optional: CloudFormation automation

If you want full automation later, we provided `infrastructure/cloudfront-response-headers.yaml` and `scripts/generate_cloudfront_policy.ps1`. Use those to produce the Response Headers Policy and then attach it to your distribution as described above.

Quick Ops Reference

For on-call checks, validation, and basic deployment tasks, see:
OPS_QUICK_REFERENCE.md

This file contains one-line verification commands and quick reload/deploy steps useful for staging and production troubleshooting.

Cloudflare Deployment (autolearnpro.com)

Purpose

Mirror the baseline headers defined in `next.config.js` at the edge via Cloudflare. Ensures catalog template security consistency across all environments.

Apply HTTP Response Headers

Condition:

* `URL matches *` (applies to all pages/resources)

Response Headers:

Header	Value
Content-Security-Policy	`default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'; img-src 'self' data:; font-src 'self'; connect-src 'self'; frame-ancestors 'none'; base-uri 'self'; form-action 'self'`
Strict-Transport-Security	`max-age=63072000; includeSubDomains; preload` *(only if HTTPS everywhere)*
Cross-Origin-Opener-Policy	`same-origin`
X-Frame-Options	`DENY`

Referrer-Policy	`no-referrer-when-downgrade`

Verification

Run the following from a terminal to confirm headers are applied:

```bash
curl -I https://autolearnpro.com/
```

Check that all headers match the table above and that there are no duplicates or conflicts.

Optional Hardening (Future)

* Permissions-Policy (camera, microphone, geolocation)
* Remove `Server` header
* Enable “Always Use HTTPS” in Cloudflare

Catalog reviewer note

This Cloudflare Response Headers configuration mirrors the application-layer headers in `frontend/web/next.config.js` and completes edge-level header coverage for baseline approval. Reviewers should confirm the CSP host lists match runtime dependencies in staging and that HSTS is enabled only after TLS termination and certificate management are validated. Once approved, ops/security can apply these Response Header Rules to `autolearnpro.com` to ensure consistent defense-in-depth across deployments.