Application CDN Git Deployment

Namazu Elements Core Features · https://namazustudios.com/docs/namazu-elements-core/application-cdn-git-deployment/

Namazu Elements includes a git-push deployment pipeline for publishing an entire static site (a single-page app, a marketing page, downloadable assets, and so on) under an Application, independent of any single Element. You push a git repository, tell Elements which revision to publish, and it serves that revision's file tree over HTTP.

> Note
>
> This is a different system from Element Static Content and Dashboard UI Plugins (which serves files bundled inside a specific Element's `.elm` archive) and from the Large Object API (which stores individual files as first-class database records). All three happen to live under a `/cdn`-ish or `/app`-ish URL depending on the system, but they are separate storage mechanisms serving different use cases.

How It Works

1. You push a git repository containing your static site to an Application-scoped git endpoint over HTTP.
2. You call the Deployment REST API with the git commit SHA (the "revision") you want to publish.
3. Elements checks out that revision's file tree into local storage and atomically points a version symlink at it.
4. Requests to the serving endpoint for that Application and version are answered from the checked-out tree.

Pushing Content

Every Application has its own git remote at `/cdn/git/{applicationName}.git`, backed by a standard git-over-HTTP smart transport and protected with HTTP Basic authentication. Point a normal git remote at it and push:

```
git remote add elements-cdn https://{username}:{password}@{host}/cdn/git/{applicationName}.git
git push elements-cdn main
```

Pushing content does not publish it by itself; it only makes that commit available for Elements to check out. Publishing happens in the next step.

Publishing a Deployment

A **Deployment** ties a version label to a git revision for a given Application. Deployments are managed through the Deployment REST resource:

Method	Path	Description
`POST`	`/deployment/{applicationId}`	Creates a new Deployment for the Application from a given `revision` (git commit SHA).
`PUT`	`/deployment/{applicationId}/{version}`	Updates an existing version to point at a new `revision`.
`GET`	`/deployment/{applicationId}`	Gets the Application's current Deployment.
`DELETE`	`/deployment/{applicationId}/{version}`	Removes a Deployment and its checked-out content.

A Deployment record has an `id`, a `version`, the `revision` it was published from, and the owning `application`. Creating or updating a Deployment (and deleting one) requires Superuser access. Unauthenticated callers may only read the current Deployment; they cannot create, update, or delete one.

How Content Is Stored and Served

When a Deployment is created or updated, Elements clones the repository content at the given revision into a uniquely named directory, then atomically creates (or repoints) a symlink named after the version at the serving location:

```
{storage.directory}/{applicationName}/{clone.endpoint}/{uuid}/...          <- checked-out
revision content
{storage.directory}/{applicationName}/{serve.endpoint}/{version} -> ...{uuid}   <- symlink used
for serving
```

Deleting a Deployment removes both the symlink and the underlying checked-out directory tree. Published content is served publicly at:

```
/cdn/static/app/{applicationName}/{serve.endpoint}/{version}/{path...}
```

The serving endpoint resolves the Application from the URL, refuses to serve any path that would resolve outside that Application's own content directory, and supports conditional requests (`ETag` / `If-None-Match`) with a configurable public cache lifetime.

Configuration

Attribute	Default	Purpose
`dev.getelements.elements.cdnserve.storage.directory`	`content`	Filesystem directory used to store checked-out and served content per Application.
`dev.getelements.elements.cdnserve.endpoint.clone`	`clone`	Subdirectory name used for checked-out revision content.
`dev.getelements.elements.cdnserve.endpoint.serve`	`serve`	Subdirectory name used for the version symlinks that are actually served.
`dev.getelements.elements.git.cdn.storage.directory`	`cdn-repos/git`	Filesystem directory used to store the bare git repositories pushed to `/cdn/git/*`.
`dev.getelements.elements.cdn.public.max.age`	`300`	Cache lifetime, in seconds, sent as `Cache-Control: public, max-age=...` for served content. Shared with the Large Object API's serving endpoint.