GitSyncMirrorService

September 4, 2026

GitSyncMirrorService

Kind: Service

Source: `atloria-monorepo/apps/api/src/git-sync/mirror.service.ts`

GitSyncMirrorService — B1's OUTBOUND lane: project substrate `main` → the customer's
GitHub repo, via the existing GitHub App (NOT a hosted git remote — the RWO-PVC repo is
never exposed through a new authenticated surface).

Semantics:

- Substrate main stays AUTHORITATIVE; the customer branch is a mirror of CONTENT, not
  history: one Git Data API commit ONTO THEIR TIP per sync, carrying an `Atloria-Source-Commit: <substrate sha>` trailer (echo guard a + audit link).
- Reads the substrate at a PINNED sha (`git show` / `ls-tree` — read-only plumbing that
  never touches the shared working tree or index) so it can run WITHOUT the per-project
  commit lock, concurrent with the committer.
- Mirrors every served page `docs/<rest>` → `<prefix>/<rest>` PLUS the `.atloria/` identity manifests → `<prefix>/.atloria/*.json`. The manifests are what let the INBOUND
  ingest map paths → documents (incl. renames via previousSlugs) from the GitHub API
  alone, with no PVC access.
- Only paths the mirror OWNS are ever deleted (managed *.md under the prefix + the two
  manifests); customer files alongside are never touched.

- FORCE-PUSH RECOVERY: when the previous mirror commit is no longer an ancestor of the
  branch tip (their history was rewritten), the full content is re-mirrored onto the new tip and the recovery is RECORDED (forcePushRecoveredAt). The customer repo is NEVER
  force-pushed (updateRef is hard-wired fast-forward).

`GitSyncMirrorService` implements B1's outbound synchronization lane, mirroring a project's authoritative substrate `main` content into the customer's GitHub repository through the existing GitHub App. It reads a pinned substrate SHA without touching the shared working tree, writes a single Git Data API commit onto the customer branch tip, and preserves customer-owned files.

The service mirrors served documentation pages and `.atloria` identity manifests, detects rewritten customer history, and performs a full content recovery without ever force-pushing the customer repository.

Methods

Method	Signature	Returns	Description
`mirrorProject`	`mirrorProject(projectId: string)`	`Promise<MirrorResult>`	Mirror one project.

Dependencies

- `PrismaService`
- `DocsRepoService`
- `GithubSyncAdapter`

Where it refuses work

- `GitSyncMirrorService` stops the work with `Error` when `!tip`.
- `GitSyncMirrorService` stops the work with an early return when `!sync || sync.direction === 'off'`.
- `GitSyncMirrorService` stops the work with an early return when `sync.provider !== 'github'`.
- `GitSyncMirrorService` stops the work with an early return when `!head`.
- `GitSyncMirrorService` stops the work with an early return when `sync.lastSourceSha === head`.

When something fails

- `GitSyncMirrorService` handles failure in 1 place: it lets it reach the caller in all 1.

Diagram

```
sequenceDiagram
    participant Caller as Sync Orchestrator
    participant Mirror as GitSyncMirrorService
    participant Substrate as Project Substrate (PVC)
    participant GitHub as Customer GitHub Repository

    Caller->>Mirror: mirrorProject(project, pinnedSha)
    Mirror->>Substrate: git show / ls-tree at pinned SHA
    Substrate-->>Mirror: docs content + .atloria manifests

    Mirror->>GitHub: Read customer branch tip
    Mirror->>GitHub: Find previous Atloria mirror commit

    alt Previous mirror commit is ancestor of tip
        Mirror->>GitHub: Create tree with managed changes only
    else Customer history was force-pushed
        Mirror->>GitHub: Rebuild full managed content onto new tip
        Mirror->>Mirror: Record forcePushRecoveredAt
    end

    Mirror->>GitHub: Create commit with Atloria-Source-Commit trailer
    Mirror->>GitHub: Fast-forward update branch ref
    GitHub-->>Mirror: Updated mirror branch
    Mirror-->>Caller: MirrorResult
```

Usage

```
import { Injectable } from '@nestjs/common';
import { GitSyncMirrorService } from './git-sync/mirror.service';

@Injectable()
export class ProjectSyncJob {
  constructor(private readonly mirrorService: GitSyncMirrorService) {}

  async syncProjectToGitHub(projectId: string, substrateMainSha: string) {
    const result = await this.mirrorService.mirrorProject({
      projectId,
      sourceSha: substrateMainSha,
    });

    if (result.forcePushRecoveredAt) {
      console.warn(
```

```
        `Recovered mirror after customer history rewrite: ${result.forcePushRecoveredAt}`,
      );
    }

    return result;
  }
}
```

AI Coding Instructions

- Keep substrate reads pinned to the requested source SHA; use read-only Git plumbing and never mutate the shared working tree or index.
- Mirror only service-owned paths: managed Markdown files beneath the configured prefix plus `.atloria/*.json` manifests. Never delete adjacent customer files.
- Create one Git Data API commit per sync on top of the customer's current branch tip, including the `Atloria-Source-Commit: <sha>` trailer.
- Treat customer branch history rewrites as recovery scenarios: rebuild managed content on the new tip and record `forcePushRecoveredAt`; never force-push customer refs.
- Preserve manifest generation and synchronization behavior, since inbound ingestion relies on `.atloria` metadata to map GitHub paths and renames back to documents.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocsRepoService`
- DEPENDS_ON → `GithubSyncAdapter`

Referenced By

- `GitSyncModule` (MODULE_PROVIDES)
- `GitSyncModule` (MODULE_EXPORTS)