BundleService

September 4, 2026

BundleService

Kind: Service

Source: `atloria-monorepo/apps/api/src/docs-repo/bundle.service.ts`

BundleService — S2.5 disaster-recovery bundles for the docs git substrate.

Since S2 the per-project repos hold NON-rebuildable state (CR branches, human-authored history) — main alone is a projection of Postgres, but the branches are not. PVC loss must therefore be survivable: once a day (worker only) every repo under DOCS_REPO_ROOT is packed
with `git bundle create <tmp> --all` and shipped to the private Azure Blob container 'docs-repo-bundles' at /.bundle, keeping the newest 7 per project.

Restore path (tested in bundle.service.spec.ts, operated via tasks/docs-repo-bundle-restore-runbook.md): a bundle is a fetchable single-file remote — clone/fetch it back onto a fresh PVC and every ref + the full history is intact.

Bundles are best-effort by design: if Azure storage is not configured the sweep warns ONCE and skips; one project's failure never stops the loop; nothing here can crash the worker or fail boot (the blob client is built lazily, never in the constructor).

`BundleService` creates daily disaster-recovery Git bundles for every project repository under `DOCS_REPO_ROOT`. Running in the worker process, it packages all refs and history with `git bundle create --all`, uploads bundles to the private `docs-repo-bundles` Azure Blob container, and retains the newest seven bundles per project.

Bundle creation is best-effort: missing Azure configuration causes a one-time warning and skips the sweep, while failures for individual projects are isolated so they cannot crash the worker or block other repositories.

Methods

Method	Signature	Returns	Description
`sweep`	`sweep()`	`Promise<void>`	Cron entrypoint — 3AM (quiet hour), gated like the outbox poller so only the dedicated worker

Method	Signature	Returns	Description
			deployment bundles, and default-on via DOCS_REPO_BUNDLES (set '…
`runOnce`	`runOnce()`	`Promise<BundleSweepResult>`	One full sweep: bundle + upload every repo under DOCS_REPO_ROOT, prune retention.
`buildContainerClient`	`buildContainerClient()`	`ContainerClient	null`	

Dependencies

- `ConfigService`

Where it refuses work

- `BundleService` stops the work with an early return when `this.config.get<string>('DOCS_REPO_WORKER') !== 'true'`.
- `BundleService` stops the work with an early return when `this.config.get<string>('DOCS_REPO_BUNDLES') === 'false'`.
- `BundleService` stops the work with an early return when `!container`.
- `BundleService` stops the work with an early return when `this.containerClient !== undefined`.
- `BundleService` stops the work with an early return when `!accountName`.

When something fails

- `BundleService` handles failure in 3 places: it logs it and continues in 2, and turns it into a return value in 1.

Diagram

```
sequenceDiagram
    participant Worker
    participant BundleService
    participant Filesystem as DOCS_REPO_ROOT
    participant Git
    participant Azure as Azure Blob Storage

    Worker->>BundleService: sweep()
    BundleService->>BundleService: Build blob client lazily

    alt Azure storage is not configured
        BundleService->>Worker: Warn once and skip
    else Storage is configured
        BundleService->>Filesystem: Discover project repositories
```

```
        loop Each project repository
            BundleService->>Git: git bundle create <tmp> --all
            Git-->>BundleService: Bundle file containing all refs/history
            BundleService->>Azure: Upload <projectId>/<yyyy-mm-dd>.bundle
            BundleService->>Azure: Delete bundles beyond newest 7

            alt Project bundle fails
                BundleService->>BundleService: Log failure and continue
            end
        end

        BundleService-->>Worker: Sweep complete
    end
```

Usage

```
import { BundleService } from './bundle.service';

// Typically invoked by a scheduled worker job.
export class DocsRepoMaintenanceJob {
  constructor(private readonly bundleService: BundleService) {}

  async runDailyBundleSweep(): Promise<void> {
    // Best-effort scheduled execution. Configuration or per-project failures
    // are handled internally and should not fail the worker process.
    await this.bundleService.sweep();
  }

  async inspectBundleRun() {
    // Useful for jobs, diagnostics, or tests that need sweep results.
    const result = await this.bundleService.runOnce();

    console.log('Bundle sweep result:', result);
    return result;
  }
}
```

AI Coding Instructions

- Keep all Azure Blob client initialization lazy; do not create or validate the storage client in the constructor, because unavailable bundle storage must never prevent worker boot.
- Preserve best-effort behavior: catch and log failures per project so one corrupt repository, Git failure, or upload failure does not stop the remaining sweep.
- Create bundles with `git bundle create <temporary-path> --all`; bundles must include every ref and complete history, not only the `main` branch.
- Store uploads using the established `<projectId>/<yyyy-mm-dd>.bundle` naming convention in the private `docs-repo-bundles` container, and retain only the newest seven bundles per project.

- When changing restore or bundle semantics, update the associated tests and the `tasks/docs-repo-bundle-restore-runbook.md` operational procedure.

Relationships

- DEPENDS_ON → `configservice`

Referenced By

- `DocsRepoModule` (MODULE_PROVIDES)