TechnicalDocsQueue

September 4, 2026

TechnicalDocsQueue

Kind: Service

Source: `atloria-monorepo/apps/api/src/technical-docs/technical-docs-queue.service.ts`

Queues technical-docs generation OFF the api request path (BullMQ, like the parsing queue).

The api pods only ENQUEUE. The heavy clone+parse (several GB for a big repo) runs in the WORKER,
which is gated by TECHNICAL_DOCS_WORKER=true — so it only spins up in a dedicated worker
deployment (sized for parsing, like atlorix-runner), never in the request-serving pods. If a
parse OOMs, only that job fails/retries; the api keeps serving. Concurrency is bounded so a
couple of large parses can't pile up.

`TechnicalDocsQueue` moves technical-documentation generation off the API request path by enqueueing BullMQ jobs rather than cloning and parsing repositories synchronously. API pods only submit work; a dedicated worker deployment, enabled with `TECHNICAL_DOCS_WORKER=true`, performs the memory-intensive clone and parse operation with bounded concurrency and isolated retry failures.

Methods

Method	Signature	Returns	Description
`onModuleInit`	`onModuleInit()`	`unknown`	
`enqueue`	`enqueue(data: TechnicalDocsQueueJob)`	`Promise<string`	undefined>`
`onModuleDestroy`	`onModuleDestroy()`	`unknown`	

Dependencies

- `ConfigService`
- `TechnicalDocsGenerationService`
- `TechDocsRagService`
- `TechnicalDocsEnrichmentService`
- `TechnicalDocsMaterializerService`
- `PrismaService`
- `OpsAlertService`
- `AIUsageService`

Where it refuses work

- `TechnicalDocsQueue` stops the work with an early return when `!jobId`.

When something fails

- `TechnicalDocsQueue` handles failure in 2 places: it logs it and continues in 1, and lets it reach the caller in 1.

Diagram

```
sequenceDiagram
    participant Client
    participant API as API Pod
    participant Queue as BullMQ Queue
    participant Worker as Technical Docs Worker
    participant Repo as Git Repository
    participant Docs as Generated Docs Storage

    Client->>API: Request technical docs generation
    API->>Queue: Enqueue generation job
    API-->>Client: Return accepted response

    Note over Worker: Runs only when<br/>TECHNICAL_DOCS_WORKER=true
    Worker->>Queue: Claim job (bounded concurrency)
    Worker->>Repo: Clone repository
    Worker->>Worker: Parse source and generate docs
    Worker->>Docs: Persist generated documentation
    Worker->>Queue: Mark job complete

    Note over Queue,Worker: Failed/OOM jobs are retried<br/>without affecting API availability
```

Usage

```
import { Injectable } from '@nestjs/common';
import { TechnicalDocsQueue } from './technical-docs-queue.service';

@Injectable()
export class RepositoryService {
  constructor(
    private readonly technicalDocsQueue: TechnicalDocsQueue,
  ) {}

  async requestTechnicalDocs(repositoryId: string, branch = 'main') {
    const job = await this.technicalDocsQueue.enqueue({
      repositoryId,
      branch,
    });

    return {
      jobId: job.id,
      status: 'queued',
    };
  }
}
```

AI Coding Instructions

- Enqueue technical-docs work from request-facing services; do not clone, parse, or generate documentation directly in API handlers.
- Keep worker processing gated behind `TECHNICAL_DOCS_WORKER=true` so heavy repository operations run only in the dedicated worker deployment.
- Configure bounded BullMQ concurrency and retries carefully; large repositories can consume several GB of memory.
- Ensure queued job payloads contain stable identifiers and minimal metadata needed for the worker to reload repository state.
- Treat job failures as asynchronous outcomes: surface job status/errors through persisted state or queue inspection rather than failing the original API request.

Relationships

- DEPENDS_ON → `configservice`
- DEPENDS_ON → `TechnicalDocsGenerationService`
- DEPENDS_ON → `TechDocsRagService`
- DEPENDS_ON → `TechnicalDocsEnrichmentService`

- DEPENDS_ON → `TechnicalDocsMaterializerService`
- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `OpsAlertService`
- DEPENDS_ON → `AIUsageService`

Referenced By

- `TechDocsSourcesService` (DEPENDS_ON)
- `TechnicalDocsMcpController` (DEPENDS_ON)
- `TechnicalDocsController` (DEPENDS_ON)
- `TechnicalDocsModule` (MODULE_PROVIDES)
- `TechnicalDocsModule` (MODULE_EXPORTS)
- `TrialService` (DEPENDS_ON)