CollaborationMetricsService

September 4, 2026

CollaborationMetricsService

Kind: Service

Source: `atloria-monorepo/apps/api/src/collaboration/collaboration-metrics.service.ts`

Service for tracking and reporting collaboration metrics

`CollaborationMetricsService` is a NestJS service that collects and reports operational metrics for real-time collaboration, including active session counts and synchronization latency. It manages periodic metric recording during the module lifecycle and exposes query methods for current, aggregate, and P95 latency values.

Methods

Method	Signature	Returns	Description
`onModuleInit`	`onModuleInit()`	`void`	Initialize cleanup interval on module start
`onModuleDestroy`	`onModuleDestroy()`	`void`	Clear cleanup interval on module destroy
`recordMetrics`	`recordMetrics(documentId: string, metrics: Partial<Omit<CollaborationMetrics, 'documentId'>>)`	`Promise<void>`	Record metrics for a document
`recordSyncLatency`	`recordSyncLatency(documentId: string, latencyMs: number)`	`Promise<void>`	Record a sync latency sample
`recordActiveSessions`	`recordActiveSessions(documentId: string, count: number)`	`Promise<void>`	Record active session count
`getMetrics`	`getMetrics(documentId: string)`	`Promise<CollaborationMetrics | null>`	

Method	Signature	Returns	Description
`getActiveSessions`	`getActiveSessions(documentId: string)`	`Promise<number>`	Get active sessions count for a document
`getSyncLatency`	`getSyncLatency(documentId: string)`	`Promise<number>`	Get sync latency for a document (average)
`getP95SyncLatency`	`getP95SyncLatency(documentId: string)`	`Promise<number>`	Get p95 sync latency
`getTotalActiveSessions`	`getTotalActiveSessions()`	`Promise<number>`	Get total active sessions across all cached documents
`getGlobalAverageSyncLatency`	`getGlobalAverageSyncLatency()`	`Promise<number>`	Get average sync latency across all cached documents
`clearMetrics`	`clearMetrics(documentId: string)`	`Promise<void>`	Clear metrics for a document
`getMetricsSummary`	`getMetricsSummary()`	`Promise<{`	

```
totalActiveDocuments: number;
totalActiveSessions: number;
averageSyncLatencyMs: number;
documentsAboveLatencyThreshold: number;
```

}>` | Get a summary of collaboration metrics for monitoring |

Dependencies

- `RedisService`

Where it refuses work

- `CollaborationMetricsService` stops the work with an early return when `cached && cached.expiresAt > Date.now()`, in 4 places.
- `CollaborationMetricsService` stops the work with an early return when `samples.length === 0`.

- `CollaborationMetricsService` stops the work with an early return when `numbers.length === 0`.

When something fails

- `CollaborationMetricsService` handles failure in 6 places: it logs it and continues in 3, and turns it into a return value in 3.

Diagram

```
sequenceDiagram
  participant Nest as NestJS Lifecycle
  participant Service as CollaborationMetricsService
  participant Sessions as Collaboration Sessions
  participant Metrics as Metrics Store

  Nest->>Service: onModuleInit()
  Service->>Service: Start periodic metric recording

  loop Recording interval
    Service->>Sessions: Read active sessions
    Sessions-->>Service: Session count
    Service->>Sessions: Read sync latency samples
    Sessions-->>Service: Latency values
    Service->>Metrics: Store collaboration metrics
  end

  Client->>Service: getMetrics()
  Service->>Metrics: Retrieve latest metrics
  Metrics-->>Service: CollaborationMetrics | null
  Service-->>Client: Metrics response

  Nest->>Service: onModuleDestroy()
  Service->>Service: Stop periodic recording
```

Usage

```
import { Injectable } from '@nestjs/common';
import { CollaborationMetricsService } from './collaboration-metrics.service';

@Injectable()
export class CollaborationHealthService {
  constructor(
    private readonly collaborationMetrics: CollaborationMetricsService,
  ) {}

  async getHealthSummary() {
    const [metrics, activeSessions, p95SyncLatency] = await Promise.all([
      this.collaborationMetrics.getMetrics(),
      this.collaborationMetrics.getTotalActiveSessions(),
      this.collaborationMetrics.getP95SyncLatency(),
    ]);

    return {
      activeSessions,
```

```
      p95SyncLatencyMs: p95SyncLatency,
      latestMetrics: metrics,
      healthy: p95SyncLatency < 500,
    };
  }
}
```

AI Coding Instructions

- Let NestJS manage lifecycle hooks; keep initialization logic in `onModuleInit()` and always release timers, subscriptions, or resources in `onModuleDestroy()`.
- Use `recordMetrics()` as the orchestration point for collection work, delegating session and latency collection to `recordActiveSessions()` and `recordSyncLatency()`.
- Treat `getMetrics()` as nullable and handle the case where no metric snapshot has been recorded yet.
- Preserve the distinction between current latency (`getSyncLatency()`) and tail latency (`getP95SyncLatency()`); use P95 for health checks and alerting.
- Avoid performing expensive aggregation work in request paths; record and cache metric values on the service's scheduled collection cycle.

Relationships

- DEPENDS_ON → `RedisService`

Referenced By

- `CollaborationModule` (MODULE_PROVIDES)
- `CollaborationModule` (MODULE_EXPORTS)