GitService

September 4, 2026

GitService

Kind: Service

Source: `atloria-monorepo/apps/code-analyzer/src/app/git/git.service.ts`

Git service for cloning repositories with mono-repo support

Features:

- Shallow clones for speed (depth=1 by default)
- Support for specific branches
- Mono-repo folder path support
- Automatic cleanup of old clones
- Git authentication via tokens (for private repos)

`GitService` is a NestJS backend service responsible for cloning Git repositories to a local workspace for analysis, with built-in support for mono-repo subfolder targeting. It optimizes cloning via shallow clones (default `depth=1`), supports selecting specific branches, and handles authentication for private repositories via tokens. It also manages lifecycle concerns like cleaning up old clones to keep disk usage under control.

Methods

Method	Signature	Returns	Description
`clone`	`clone(options: CloneOptions)`	`Promise<CloneResult>`	Clone a Git repository with optional mono-repo folder support
`cleanup`	`cleanup(clonePath: string)`	`Promise<void>`	Cleanup cloned repository

Method	Signature	Returns	Description
`cleanupOld`	`cleanupOld()`	`Promise<void>`	Cleanup all old clones (older than 1 hour)
`checkGitInstalled`	`checkGitInstalled()`	`Promise<boolean>`	Check if Git is installed

Where it refuses work

- `GitService` stops the work with `Error` when `typeof branch !== 'string' || !/^[A-Za-z0-9][\w./+@-]{0,254}$/.test(branch)`.
- `GitService` stops the work with `Error` when `u.protocol !== 'https:' && u.protocol !== 'http:'` — “Only http(s) repository URLs are supported”.

When something fails

- `GitService` handles failure in 7 places: it logs it and continues in 3, lets it reach the caller in 3, and turns it into a return value in 1.

Diagram

```
sequenceDiagram
  autonumber
  actor Caller as Analyzer/Controller
  participant GitService as GitService
  participant FS as Local FS
  participant Git as Git CLI/Library
  participant Cleanup as Cleanup Job

  Caller->>GitService: clone({ repoUrl, branch?, token?, depth=1, monoRepoPath? })
  GitService->>Cleanup: cleanupOldClones()
  Cleanup->>FS: delete stale clone dirs
  GitService->>Git: git clone --depth=1 (--branch <branch>) <repoUrl> <targetDir>
  alt Private repo (token provided)
    GitService->>Git: inject auth (token in URL/headers)
  end
  Git->>FS: write working tree to <targetDir>
  alt monoRepoPath provided
    GitService->>FS: resolve <targetDir>/<monoRepoPath>
    GitService-->>Caller: return path to mono-repo subfolder
  else
    GitService-->>Caller: return path to repo root
  end
```

Usage

```
import { Injectable } from '@nestjs/common';
import { GitService } from './git.service';

@Injectable()
export class RepoAnalysisService {
  constructor(private readonly git: GitService) {}

  async analyze() {
    const localPath = await this.git.clone({
      repoUrl: 'https://github.com/acme/atloria-monorepo.git',
      branch: 'main',
      depth: 1, // shallow clone for speed (default behavior)
      monoRepoPath: 'apps/code-analyzer', // optional: target subfolder within a mono-repo
      token: process.env.GITHUB_TOKEN, // optional: required for private repos
    });

    // Use localPath as the root for scanning/analysis
    // e.g., await this.scanner.scan(localPath);
    return { localPath };
  }
}
```

AI Coding Instructions

- Prefer shallow clones (`depth=1`) unless the feature explicitly needs history; increasing depth can dramatically impact performance and disk usage.
- When adding auth changes, ensure tokens are never logged and are not written to disk (avoid persisting tokenized URLs in config files).
- Treat `monoRepoPath` as a resolved, validated subdirectory of the clone target to avoid path traversal or accidental analysis of the wrong folder.
- Keep cleanup idempotent and safe: only delete within the managed clones directory, and never remove user-provided paths outside that root.
- Integration point: callers should rely on the returned local path (repo root or mono-repo subfolder) and avoid assuming directory structure before `clone()` resolves it.

Referenced By

- `AnalysisService` (DEPENDS_ON)
- `AppService` (DEPENDS_ON)
- `GitModule` (MODULE_PROVIDES)
- `GitModule` (MODULE_EXPORTS)