

@push.rocks/smartantivirus

an antivirus package

- [readme.md for @push.rocks/smartantivirus](#)
- [changelog.md for @push.rocks/smartantivirus](#)

readme.md for @push.rocks/smartantivirus

Enterprise-grade antivirus scanning for Node.js applications - Seamlessly integrate ClamAV's powerful virus detection into your TypeScript/JavaScript projects with zero hassle.

Why SmartAntivirus? ☐☐

In today's digital landscape, security is paramount. Whether you're building a file-sharing platform, processing user uploads, or handling sensitive data streams, you need reliable virus protection that just works. SmartAntivirus gives you:

- ☐☐ **Docker-based or Direct Connection** - Choose your deployment style
- ⚡ **In-memory Scanning** - Lightning-fast scanning without disk I/O
- ☐☐ **Stream Processing** - Scan data on-the-fly as it flows through your app
- ☐☐ **TypeScript First** - Full type safety and IntelliSense support
- ☐☐ **Zero Config** - Works out of the box with sensible defaults
- ☐☐ **Auto-updating** - Virus definitions stay current automatically

Install

```
npm install @push.rocks/smartantivirus
```

Or if you're using pnpm (recommended):

```
pnpm add @push.rocks/smartantivirus
```

Quick Start ☐☐♂

The 5-Minute Setup

```
import { ClamAvService } from '@push.rocks/smartantivirus';

// That's it! The service automatically manages a Docker container
const scanner = new ClamAvService();

// Scan a suspicious string
const result = await scanner.scanString('Is this text safe?');
console.log(result.isInfected ? '⚠️ Threat detected!' : '✅ All clear!');

// Scan a buffer
const fileBuffer = await fs.readFile('./upload.pdf');
const scanResult = await scanner.scanBuffer(fileBuffer);
```

Core Concepts 📖

SmartAntivirus provides two main classes:

📖 ClamAvService

The high-level interface for virus scanning. It handles all the complexity behind a simple, intuitive API.

📖 ClamAVManager

Low-level Docker container management for advanced use cases. Most users won't need to interact with this directly.

Real-World Examples 📖

Protecting File Uploads

```
import { ClamAvService } from '@push.rocks/smartantivirus';
import express from 'express';
import multer from 'multer';
```

```
const app = express();
const scanner = new ClamAvService();
const upload = multer({ storage: multer.memoryStorage() });

app.post('/upload', upload.single('file'), async (req, res) => {
  try {
    // Scan the uploaded file buffer
    const result = await scanner.scanBuffer(req.file.buffer);

    if (result.isInfected) {
      return res.status(400).json({
        error: 'File rejected',
        threat: result.reason
      });
    }

    // File is safe, proceed with storage
    await saveFile(req.file);
    res.json({ message: 'File uploaded successfully' });
  } catch (error) {
    res.status(500).json({ error: 'Scan failed' });
  }
});
```

Streaming Large Files

Never load huge files into memory! Stream them instead:

```
import { ClamAvService } from '@push.rocks/smartantivirus';
import { createReadStream } from 'fs';

const scanner = new ClamAvService();

async function scanLargeFile(filePath: string) {
  const stream = createReadStream(filePath);
  const result = await scanner.scanStream(stream);

  if (result.isInfected) {
```

```
    console.log(`⚠️Threat found: ${result.reason}`);
    // Quarantine or delete the file
  } else {
    console.log('✅ File is clean');
  }
}
```

Scanning Remote Content

Perfect for proxies, CDNs, or content moderation:

```
const scanner = new ClamAvService();

// Scan a file from a URL
const result = await scanner.scanFileFromWebAsStream('https://example.com/document.pdf');

// For browser environments using Web Streams API
async function scanInBrowser(url: string) {
  const response = await fetch(url);
  const webStream = response.body as ReadableStream<Uint8Array>;

  if (webStream) {
    const result = await scanner.scanWebStream(webStream);
    return result;
  }
}
```

Advanced Container Management

For production environments requiring fine-grained control:

```
import { ClamAVManager } from '@push.rocks/smartantivirus';

class AntivirusService {
  private manager: ClamAVManager;

  async initialize() {
    this.manager = new ClamAVManager();
  }
}
```

```

// Start the container
await this.manager.startContainer();

// Set up log monitoring
this.manager.on('log', (event) => {
  if (event.type === 'error') {
    console.error(`ClamAV Error: ${event.message}`);
    // Send to your logging service
  }
});

// Update virus definitions
await this.manager.updateDatabase();

// Get database info
const dbInfo = await this.manager.getDatabaseInfo();
console.log(`Virus DB Version: ${dbInfo}`);
}

async shutdown() {
  await this.manager.stopContainer();
}
}

```

Testing

We use the industry-standard EICAR test string for verification:

```

import { ClamAvService } from '@push.rocks/smartantivirus';

const scanner = new ClamAvService();

// This is the EICAR test string - it's harmless but triggers antivirus
const EICAR = 'X50!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*';

const result = await scanner.scanString(EICAR);
console.log(result.isInfected); // true

```

```
console.log(result.reason); // 'Eicar-Test-Signature'
```

Run the test suite:

```
npm test
```

API Reference

ClamAvService

Constructor

```
new ClamAvService(host?: string, port?: number)
```

- `host` - ClamAV daemon host (default: '127.0.0.1')
- `port` - ClamAV daemon port (default: 3310)

Methods

```
scanString(text: string): Promise<ScanResult>
```

Scan a text string for threats.

```
scanBuffer(buffer: Buffer): Promise<ScanResult>
```

Scan binary data in a Buffer.

```
scanStream(stream: NodeJS.ReadableStream): Promise<ScanResult>
```

Scan a Node.js readable stream.

```
scanWebStream(stream: ReadableStream<Uint8Array>): Promise<ScanResult>
```

Scan a Web Streams API stream (browser-compatible).

```
scanFileFromWebAsStream(url: string): Promise<ScanResult>
```

Download and scan a file from a URL.

```
verifyConnection(): Promise<boolean>
```

Test the connection to ClamAV daemon.

ScanResult Type

```
interface ScanResult {
    isInfected: boolean;
    reason?: string; // Threat name if infected
}
```

ClamAVManager

Advanced container management for production deployments:

- `startContainer()` - Launch ClamAV in Docker
- `stopContainer()` - Gracefully shutdown
- `updateDatabase()` - Update virus definitions
- `getDatabaseInfo()` - Get current DB version
- `getLogs()` - Retrieve container logs
- Event: `'log'` - Real-time log streaming

Production Considerations

Performance Tips

1. **Reuse connections** - Create one `ClamAvService` instance and reuse it
2. **Stream large files** - Don't load them into memory
3. **Implement timeouts** - Protect against hanging scans
4. **Monitor logs** - Watch for database update failures

Security Best Practices

- Run ClamAV container with limited resources
- Implement rate limiting on scan endpoints
- Log all detected threats for audit trails
- Regularly update virus definitions
- Use separate containers for different environments

Deployment Options

Docker Compose

```
services:
  clamav:
    image: clamav/clamav:latest
    ports:
      - "3310:3310"
    volumes:
      - clamav-db:/var/lib/clamav
```

Kubernetes

The service automatically manages containers, but you can also deploy ClamAV separately and connect directly to the daemon.

Troubleshooting

Common Issues

Container won't start

- Ensure Docker is running
- Check port 3310 isn't already in use
- Verify sufficient disk space for virus definitions

Scans timing out

- Large files may take time - implement appropriate timeouts
- Check container resources (CPU/Memory)
- Ensure virus database is not updating

False positives

- Some packers/obfuscators trigger detection
- Whitelist known-safe patterns if needed
- Keep virus definitions updated

Contributing & Support

- [📄 Report Issues](#)
- [📄 Documentation](#)
- [📄 Discussions](#)

License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the [license](#) file within this repository.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @push.rocks/smartantivirus

2025-08-16 - 1.3.2 - fix(build)

Bump dependencies, improve test/build scripts, expand documentation and add project metadata

- Updated devDependencies: bumped @git.zone/tsbuild, @git.zone/tsbundle, @git.zone/tstest, @push.rocks/tapbundle, @types/node and typescript to newer versions.
- Updated runtime dependencies: bumped @push.rocks/smartfile, @push.rocks/smartpath and axios; tar version updated.
- Adjusted npm scripts: test now runs with '(tstest test/ --web --verbose --logfile --timeout 120)'; build and docs scripts unchanged in behavior.
- Added packageManager entry (pnpm) and a pnpm-workspace.yaml with onlyBuiltDependencies configured.
- Expanded and rewrote README with detailed quick start, examples, API reference and troubleshooting guidance.
- Added local assistant/IDE metadata and memories (.claude/settings.local.json and .serena/*) to aid development tooling and project onboarding.

2025-02-07 - 1.3.1 - fix(core)

Updated descriptions and keywords in package.json and npmextra.json. Improved README content for usage clarity.

- Revised package.json description and keywords to better represent the project's features.
- Enhanced npmextra.json with updated module attributes.
- Improved README with clearer instructions and examples for using ClamAVManager and ClamAvService.
- Fixed incorrect import path in test.clamav.manager.ts.

2025-02-05 - 1.3.0 - feat(ClamAvService)

Add support for enhanced streaming methods in ClamAvService

- Add methods to ClamAvService: scanStream for NodeJS streams, scanWebStream for Web API streams, and scanFileFromWebAsStream for fetching and scanning files from URLs.
- Update usage examples in readme for new streaming methods.

2025-02-05 - 1.2.0 - feat(ClamAvService)

Add stream scanning methods to ClamAvService

- Added scanStream method to support scanning NodeJS streams directly.
- Introduced scanWebStream method for scanning web resources as streams.
- Integrated stream scanning into existing ClamAvService class.

2025-02-03 - 1.1.2 - fix(documentation)

Update readme with additional legal and trademark information

- Added legal information related to licensing and trademarks
- Provided company details of Task Venture Capital GmbH

2025-02-03 - 1.1.1 - fix(clamav.manager)

Improve log handling and add timeout for log reception in ClamAV manager tests

- Refined the log receiving mechanism in ClamAV manager tests to use promises for better control over log receipt timing.
- Introduced a timeout mechanism in the log receiving test case to avoid indefinite waiting.
- Fixed the test case setup to accurately reflect log receipt and database information verification.

2025-02-03 - 1.1.0 - feat(ClamAvService)

Add ClamAV Manager with Docker container management capabilities.

- Introduced ClamAVManager class to manage ClamAV Docker containers.
- Implemented startContainer and stopContainer methods in ClamAVManager.
- Integrated ClamAVManager into ClamAvService for managing container lifecycle.
- Added ClamAVManager test setups and helpers in test suite.

2025-01-10 - 1.0.4 - fix(documentation)

Removed redundant conclusion section in readme.

- Removed the conclusion section from the README file for conciseness.

2025-01-10 - 1.0.3 - fix(readme)

Fix formatting errors in the README file for consistent Markdown syntax.

- Removed stray Markdown syntax in README file.

2025-01-10 - 1.0.2 - fix(documentation)

Updated README and package metadata to reflect antivirus scanning capabilities and usage instructions.

- Improved description in package.json and npmextra.json to better describe the antivirus scanning functionalities.
- Enhanced README.md to provide detailed installation and usage instructions, usage examples, error handling, and testing guidance.

2025-01-10 - 1.0.1 - Initial Release

Initial release of the project.

- Launched the first version with core features implemented