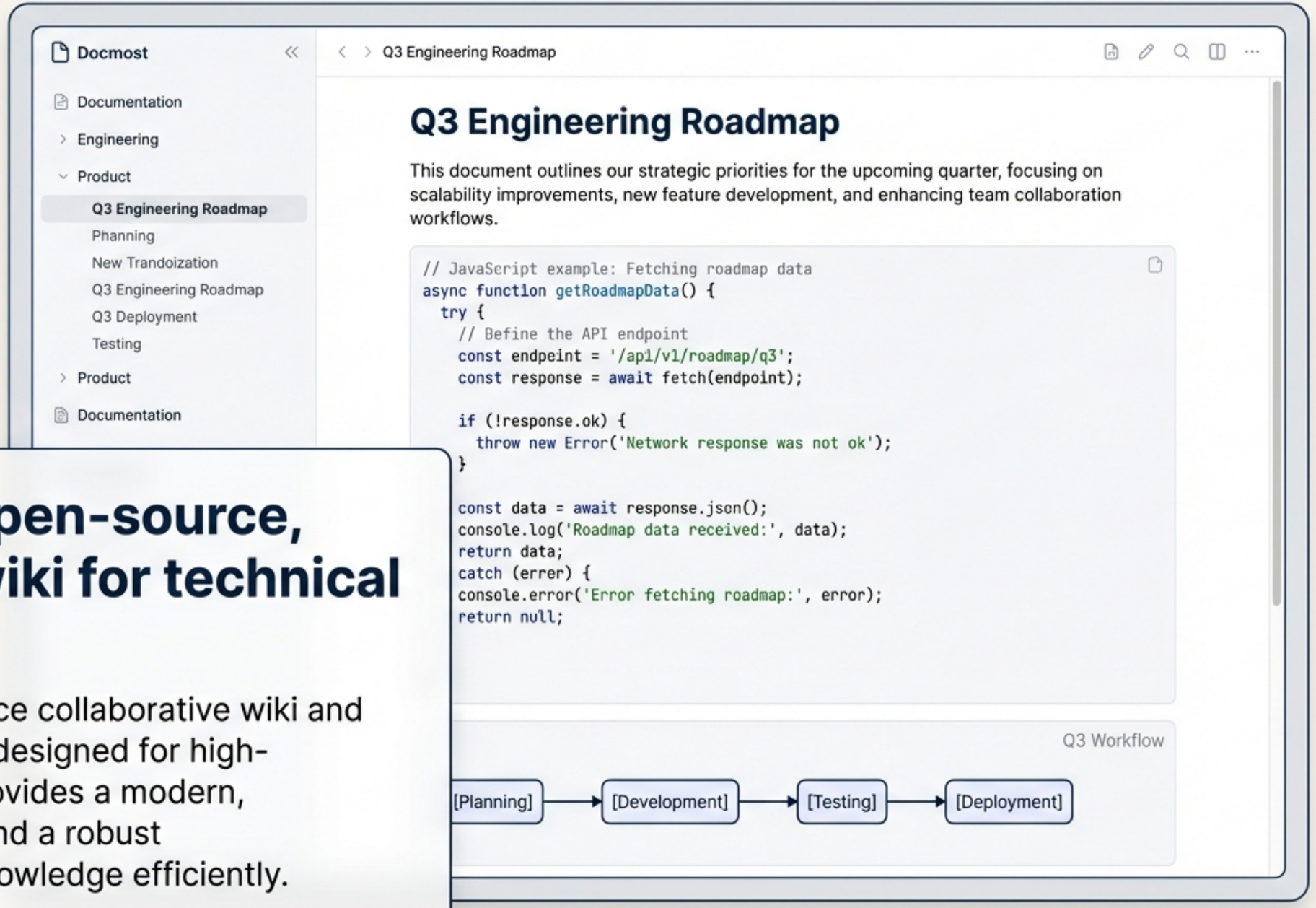


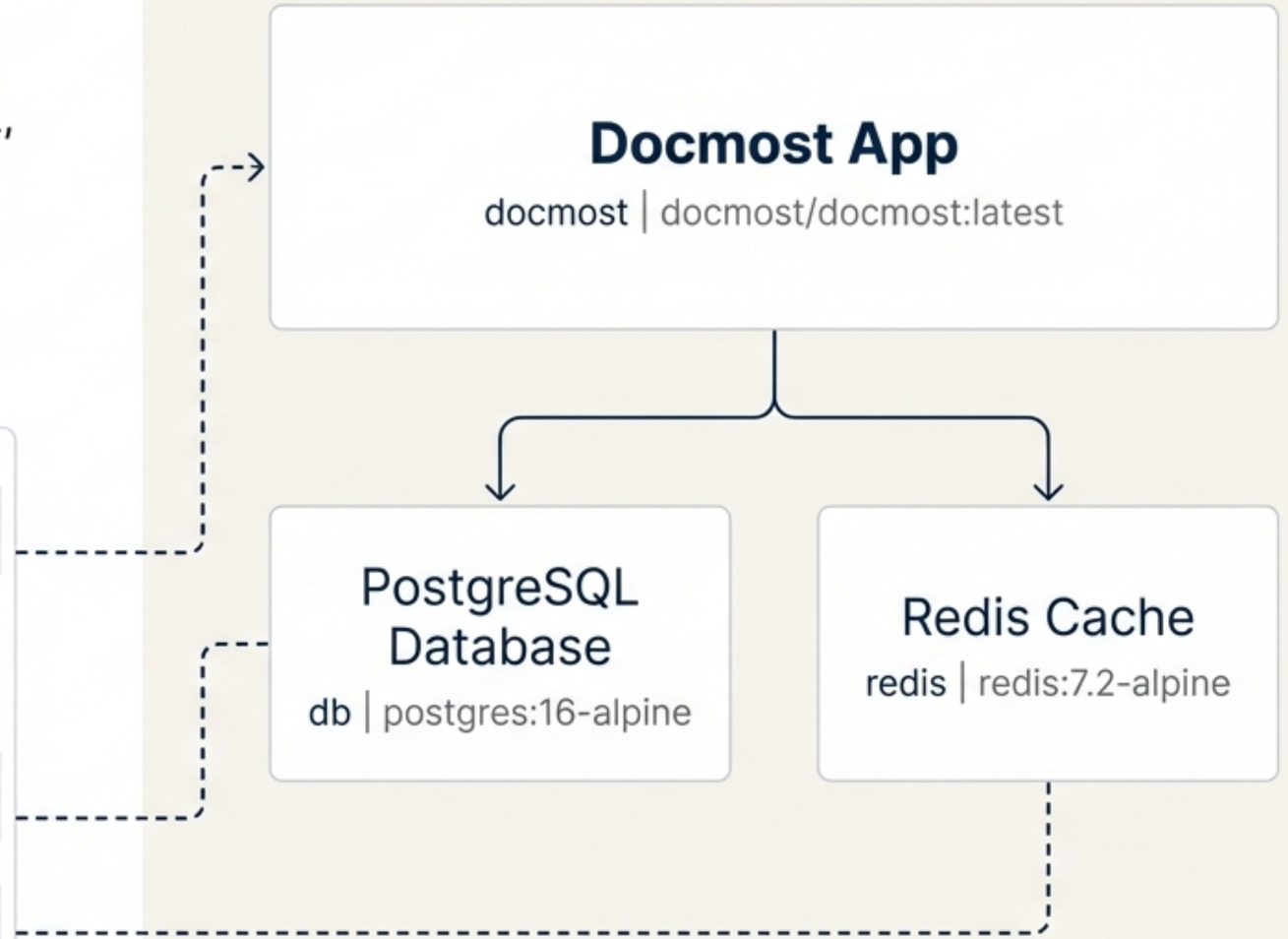
Docmost: An open-source, collaborative wiki for technical teams.

Docmost is an open-source collaborative wiki and documentation software designed for high-performance teams. It provides a modern, modern, intuitive editor and a robust backend for managing knowledge efficiently.

A Look at the Blueprint: Understanding the Core Architecture

Your first look into the repository, `docker-compose.yml`, reveals a clean, containerized architecture. The entire application stack is orchestrated through three core services, ensuring a straightforward setup and consistent development environment.

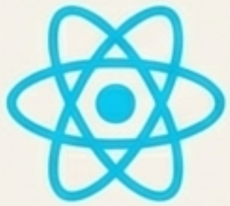
```yaml
services:
  docmost:
    image: docmost/docmost:latest
    depends_on:
      - db
      - redis
  ...
  db:
    image: postgres:16-alpine
  ...
  redis:
    image: redis:7.2-alpine
```



Built on a Modern, Performant Foundation.

Docmost leverages a full-stack TypeScript approach, combining a powerful backend framework with a reactive, component-based frontend. This choice of technologies prioritizes developer experience, performance, and type-safety.

Frontend



Backend

Collaboration

Tooling

A Scalable Monorepo Managed with Nx.

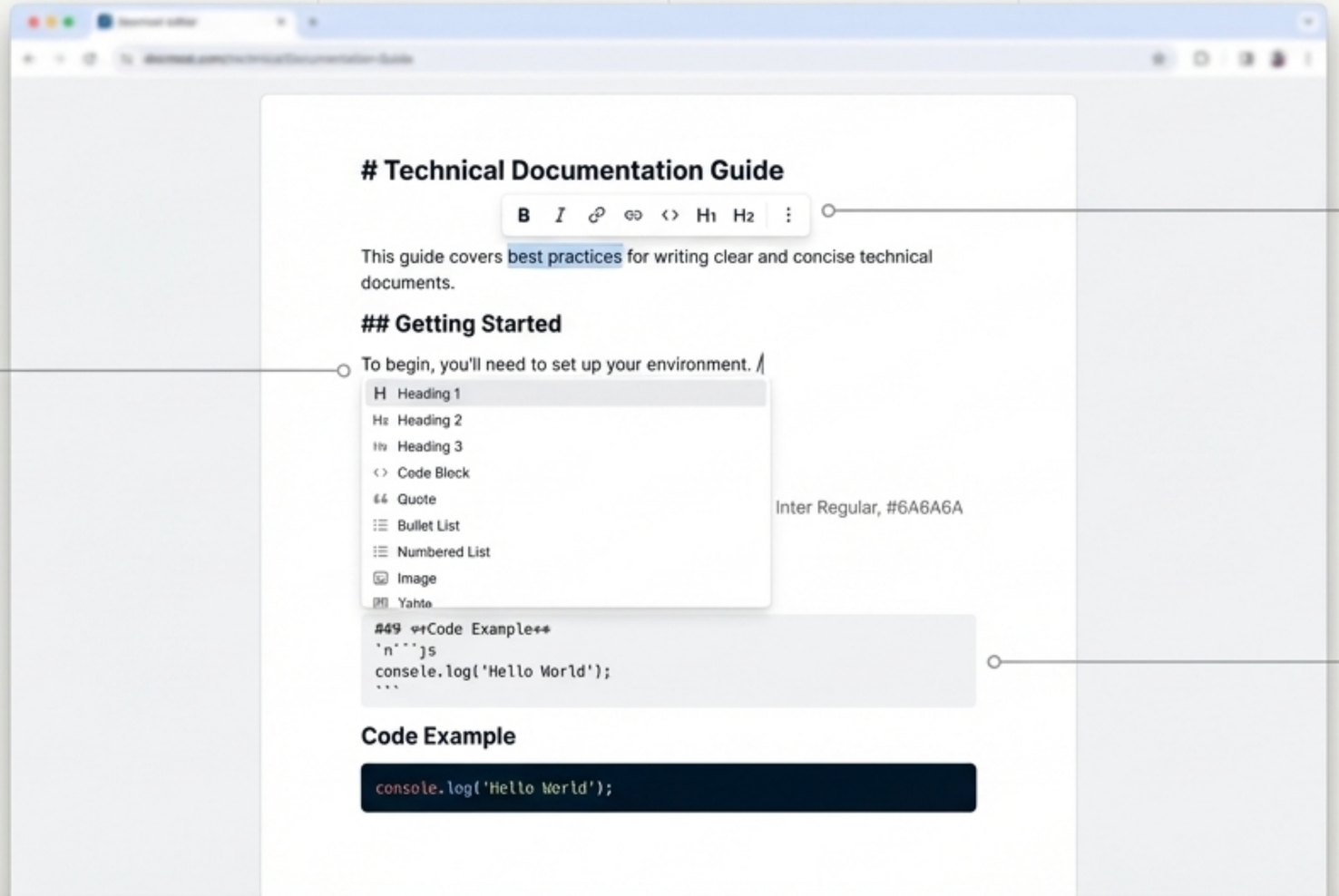
The repository is structured as an Nx monorepo, providing a clear separation of concerns between applications and shared packages. This approach enhances code sharing, streamlines build processes, and simplifies dependency management across the project.

```
{
  "targetDefaults": {
    "build": {
      "dependsOn": ["^build"],
      "cache": true
    },
    "lint": { "cache": true }
  }
}
```

```
docmost/
├── apps/
│   ├── client/  (The React Frontend)
│   └── server/  (The NestJS Backend)
└── packages/
    ├── editor-ext/  (Shared Tiptap Extensions)
    └── ee/          (Enterprise Edition Code)
```

The Heart of Docmost: A Powerful, Tiptap-Based Editor.

The user experience is centered around a rich-text editor built on Tiptap. It's designed for writing technical documentation, offering both markdown shortcuts and a powerful floating menu for seamless formatting.



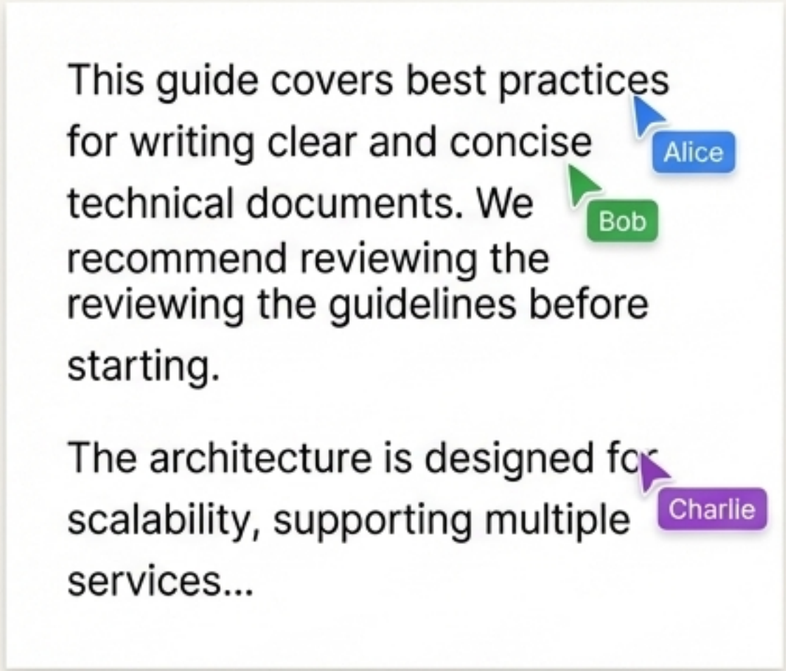
Slash commands for quick insertion (`slash-command.ts`)

Context-aware bubble menu (`bubble-menu.tsx`)

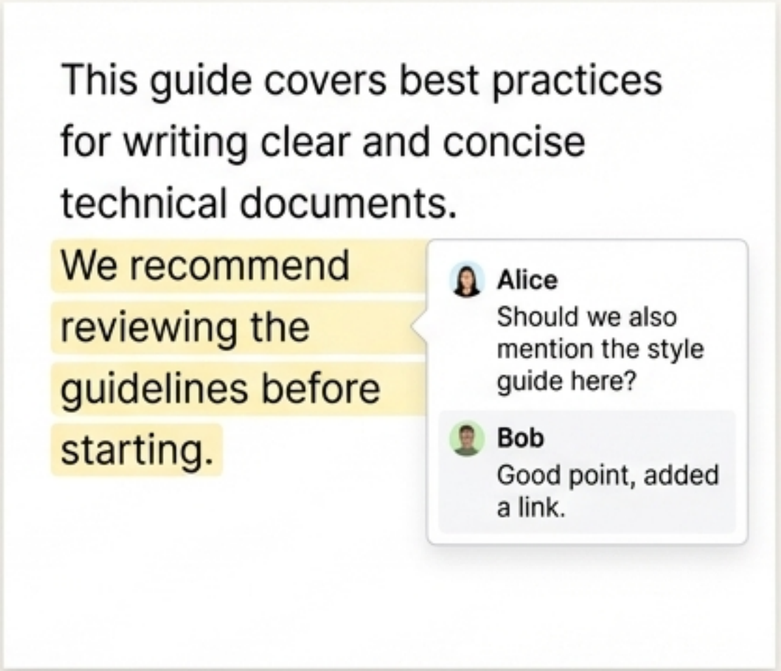
Seamless markdown paste support (`markdown-clipboard.ts`)

Beyond Static Docs: Real-time Collaboration and Context

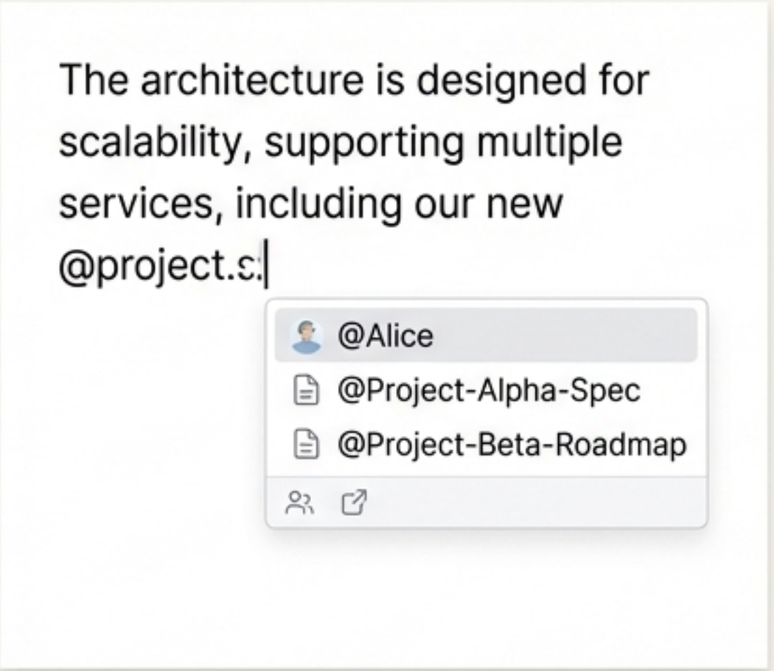
Collaboration is a first-class citizen. Real-time cursors, shared document state, and inline commenting transform documentation into a live conversation.



Live Cursors & Presence (`CollaborationCursor` extension)



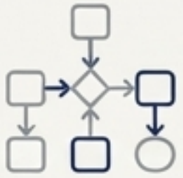
Inline Commenting (`comment-list-item.tsx`)



Mentions for Users & Pages (`mention-suggestion.ts`)

Create Rich Content with Custom Nodes and Embeds

Move beyond text and images. Embed interactive diagrams, whiteboards, and content from third-party services directly into your documents. The editor's extensibility supports a wide range of content types.

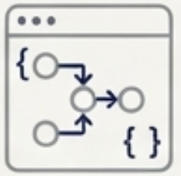


Draw.io / diagrams.net

Insert and edit complex diagrams. (`drawio.tsx`)

Excalidraw

Embed collaborative whiteboards. (`excalidraw.tsx`)



Mermaid Diagrams

Generate diagrams from code. (`slash-command.ts`)

Third-Party Embeds

Bring in content from your favorite tools. (`icons/`)

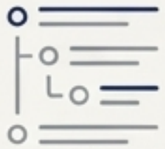


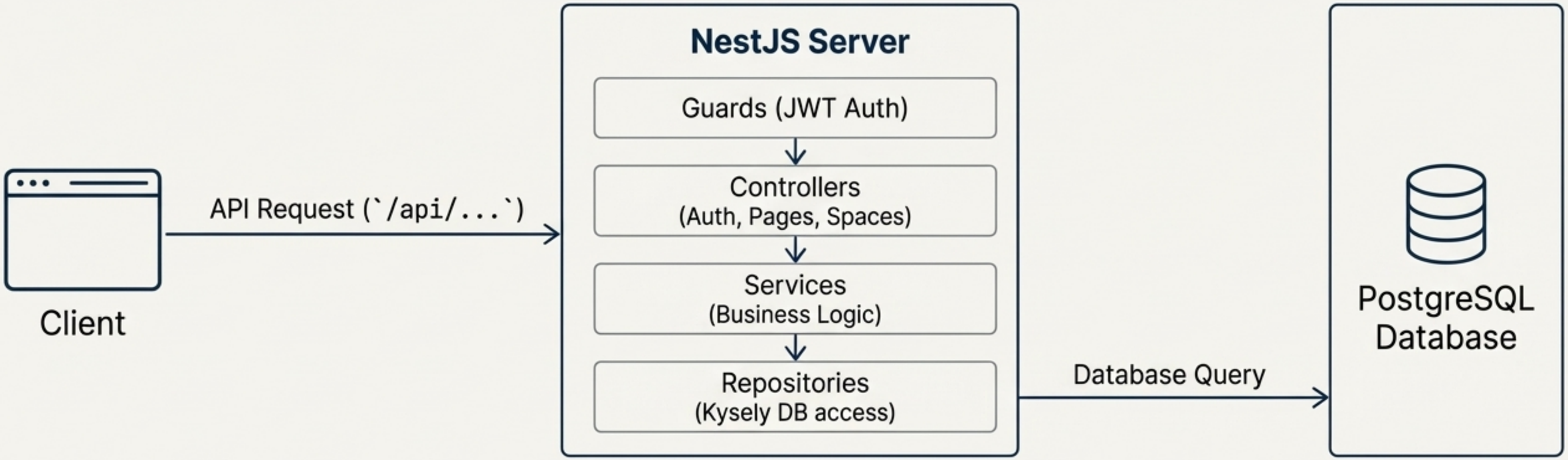
Table of Contents

Automatically generated from headings. (`table-of-contents.tsx`)

The NestJS Backend: Powering the Platform

The server is a robust NestJS application responsible for authentication, data persistence, and serving the API. It uses **Kysely** for type-safe SQL queries and **BullMQ** for managing background jobs like indexing and notifications.

```
imports: [
  AuthModule,
  UserModule,
  PageModule,
  SpaceModule
]
```



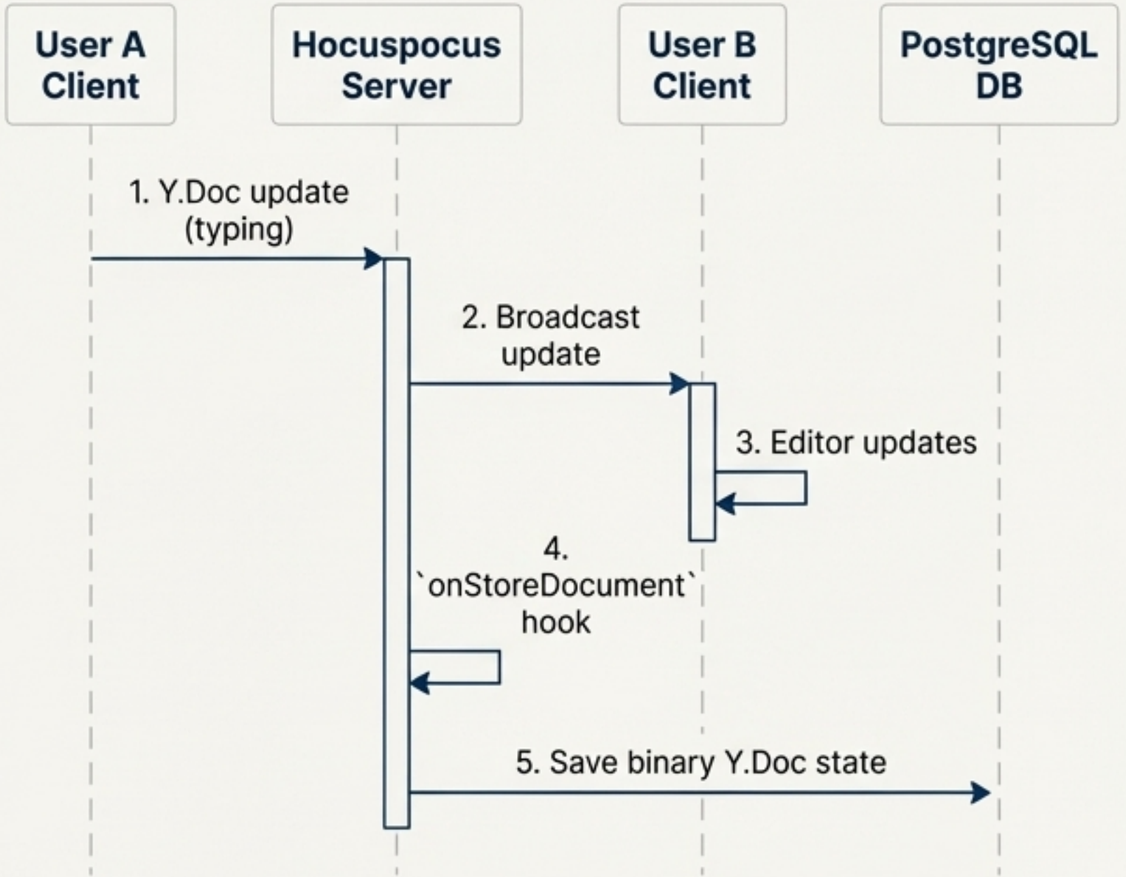
How Real-time Sync Works: Hocuspocus and WebSockets.

Real-time collaboration is powered by a Hocuspocus WebSocket server. The server's `PersistenceExtension` saves document updates (`Y.Doc` state) to the database, ensuring no data is lost. General UI updates are pushed to clients via a separate WebSocket event system.

persistence.extension.ts

```
async onStoreDocument(data: onStoreDocumentPayload) {
  // ...
  const pageId = getPageId(documentName);
  const ydoc = Y.encodeStateAsUpdate(document);

  // Save binary Y.Doc state to database
  await this.pageRepo.updatePage({ ydoc }, pageId);
  // ...
}
```



An Open Core Model: Balancing Community and Sustainability

Docmost is built on a powerful, open-source core licensed under AGPLv3. For advanced enterprise needs, a separate set of commercially licensed features is available. This model supports the project's long-term development while keeping the core functionality free and open.

Open Source Core

License:
AGPL v3

Key Features:

- Collaborative Editor
- Page & Space Management
- Core API

Enterprise Edition

License:
Docmost Enterprise License

Key Features:

- AI Search
- SSO
- Advanced Security

```
All files in the following directories are licensed under the Docmost Enterprise license...
`apps/server/src/ee`, `apps/client/src/ee`, `packages/ee`
```

Inside the Enterprise Edition.

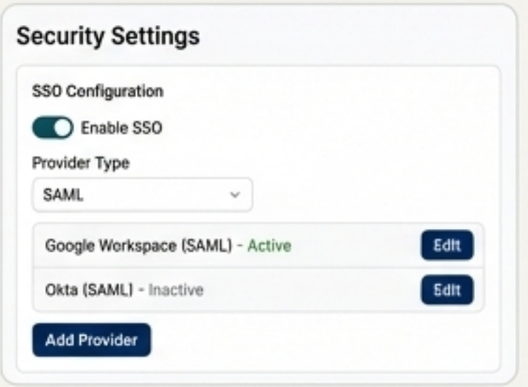
The Enterprise Edition includes features tailored for larger organizations, focusing on AI-powered productivity, enhanced security, and administrative control.

AI Search

Ask questions of your documentation and get synthesized answers with sources.

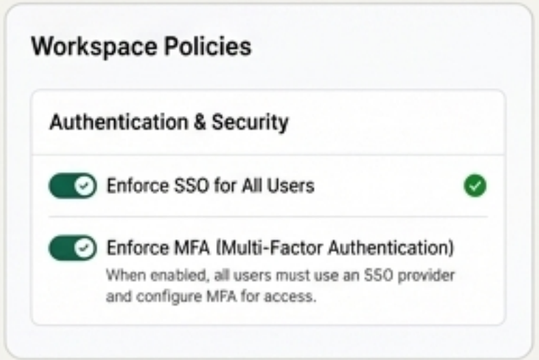
Single Sign-On (SSO)

Integrate with SAML and LDAP providers for centralized authentication.



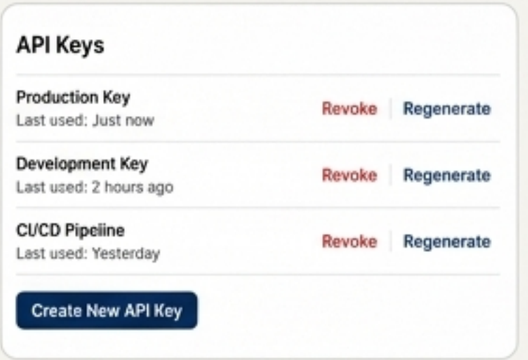
Enforce MFA & SSO

Mandate multi-factor authentication and SSO across your workspace.



REST API Keys

Generate API keys for programmatic access and integrations.



Your Turn to Build: Getting Started Locally.

Running Docmost locally is simple. Clone the repository, configure your environment, and launch the stack with Docker Compose.

1. Configure Environment

Copy `.env.example` to `.env` and fill in the required secrets.

```
APP_URL=http://localhost:3000
DATABASE_URL="postgresql://do
cmost:STRONG_DB_PASSWORD
_PASSWORD@db:5432/docmost?
schema=public"
REDIS_URL="redis://redis:6379"
```

2. Install Dependencies

```
pnpm install
```

3. Launch & Build

Access the application at `http://localhost:3000`.

```
docker-compose up -d
pnpm build
pnpm start
```

The Path to Contribution.

Contributions are welcome. Whether it's through code, documentation, or localization, you can help shape the future of Docmost.

Code

The core application is licensed under AGPLv3. Follow the development documentation for setup and contribution guidelines.

Check `README.md` for the link.

Localization

We use Crowdin to manage translations. The configuration in `crowdin.yml` points to the primary language file.

```
files:
  - source: /apps/client/public/
     locales/en-US/translation.json
    translation: /apps/client/
     public/locales/%locale%/
     %original_file_name%
```

Community

Thanks to partners like Algolia and Crowdin who support the open-source community.

algolia Crowdin

Built to Be Extended, Designed to Be Understood.

Docmost’s architecture is intentionally modular. The separation of the frontend, backend, and a dedicated `@docmost/editor-ext` package provides clear extension points. The transparent open-core model invites developers to build on a solid, collaborative foundation.