ChatGPT

On-Premise Alternatives to OneNote: Dokuwiki vs. Confluence vs. Docmost vs. BookStack vs. Nextcloud

Introduction and Evaluation Criteria

For a documentation-heavy team seeking to replace Microsoft OneNote with an **on-premise** solution, we compare five tools: **DokuWiki**, **Atlassian Confluence**, **Docmost**, **BookStack**, and **Nextcloud** (with its Notes/Collectives apps). All these can be self-hosted on VMs or Kubernetes. We evaluate them on:

- **Ease of Use for Long-Form Documentation:** Support for basic formatting (text, lists, images/screenshots, tables), UI intuitiveness for non-technical staff, and handling of very large pages or notebooks.
- **Collaborative Editing:** Real-time or concurrent editing capabilities.
- **LDAP/AD Integration:** Support for enterprise authentication (LDAP/Active Directory) and whether it's included for free or only in paid editions.
- **Attachment Support:** Ability to upload and attach files (PDFs, documents, images) to notes/pages.
- **Export Capabilities:** Options to export content (PDF, HTML, Markdown, etc.) for backup or sharing.
- **Mobile Support:** (Not required, but noted where available).
- **On-Premise Hosting Compatibility:** Ease of deploying on-premise (VMware, Docker/Kubernetes).

Finally, we highlight each tool's **strengths, weaknesses, and migration considerations** for OneNote users, especially regarding formatting habits.

Feature Comparison at a Glance

Feature/ Criteria	DokuWiki	Confluence	Docmost	BookStack	Nextcloud Collectives/ Notes
Basic Formatting (text, lists, images, tables)	**Yes:** Uses wiki markup (syntax for headings, lists, tables, etc.). WYSIWYG editing *via plugins* (not default) [1]. Supports embedded images and files via a media manager.	**Yes:** Rich WYSIWYG editor similar to MS Word. Full support for formatted text, bullet/number lists, tables, and inline images. Supports many advanced macros (for layouts, media, etc.) [2].	**Yes:** Rich text editor with Markdown shortcuts [3]. Supports headings, lists, **images**, and tables. Built-in diagram tools (Draw.io, Excalidraw, Mermaid) for visual content [4]. Modern UX akin to Notion/ Confluence [5].	**Yes:** Provides an intuitive WYSIWYG editor *and* a Markdown editor with live preview [6]. Basic styling, lists, tables, and embedded images are supported. Integrates diagrams.net for drawings/ diagrams [7].	**Yes (basic):** Uses a Markdown-based editor (Nextcloud Text). Supports bold/ italic, headings, lists, images, and by Nextcloud 24+ also tables [8] [9]. UI is clean and minimal, WYSIWYG-style editing for Markdown.

Feature/ Criteria	DokuWiki	Confluence	Docmost	BookStack	Nextcloud Collectives/ Notes
UI Ease for Non-Technical Users	Moderate – interface is simple but somewhat dated. Editing requires learning DokuWiki's syntax (unless a WYSIWYG plugin is added). Non-technical users may face a slight learning curve due to the markup and manual structuring [1] [10]. Navigation is namespace-based, which may feel less intuitive than a modern hierarchical sidebar without plugins [11].	High – polished, user-friendly interface. Non-technical users usually adapt quickly, as Confluence feels like a familiar document editor. Pages are organized in *Spaces* with a sidebar and can have nested child pages, making navigation easy. Numerous formatting buttons and macros make it straightforward to create richly formatted documentation.	High – modern and clean interface [5]. Resembles contemporary collaboration apps, so users coming from OneNote or Notion will find it intuitive. Offers "Spaces" for organization (like notebooks) [12] and a real-time editor that feels responsive. Overall designed for ease of collaboration and readability.	High – very intuitive with a "book and chapter" metaphor [13]. Non-technical staff find the library-like layout easy: content is grouped into Shelves → Books → Chapters → Pages. The sidebar and navigation are straightforward [6]. The presence of both WYSIWYG and Markdown editors means users can choose what they're comfortable with.	Moderate – the interface is simple and integrated into Nextcloud's web UI. Creating and editing pages (notes) is straightforward for basic text. However, formatting options are more limited (no fancy styling beyond Markdown basics). Navigation is via a list of pages in each "Collective" (which acts like a notebook space). Non-technical users can handle it, but it lacks the guided structure of a dedicated wiki (no multi-level page tree aside from simple parent/child pages in a Collective).

Feature/ Criteria	DokuWiki	Confluence	Docmost	BookStack	Nextcloud Collectives/ Notes
Handling Very Large Pages/ Notebooks	Generally robust for large content. Uses flat-text storage, so performance is usually good even with many pages. Extremely long single pages will render as one big HTML – usually fine, though editing a very large page in markup can be cumbersome. Supports indexes but lacks an automatic structured table of contents for the whole wiki without plugins [10]. Best for many small/ medium pages rather than one giant page (you can split content into namespaces).	Designed for enterprise scale. Can handle large knowledge bases with many spaces and pages. Individual pages can be quite large (though extremely long pages may become harder to navigate, so users often break them into sub-pages). Confluence automatically adds a page outline (table of contents) for headings if the macro is used, helping navigate large pages. Versioning and indexing scale to thousands of pages (as proven in many corporate deployments).	Built to handle extensive documentation. Organizing into *Spaces* with nested pages helps break content into manageable chunks. It supports very large pages (since it uses a modern web editor, extremely lengthy pages might load slower but there's no strict limit mentioned). Has page version history to manage frequent edits [14]. Likely capable of importing large existing docs (Notion/ Confluence import available) so it's intended for heavy content.	Can handle sizeable documentation sets via multiple Books and Chapters. Each page loads fully in the editor, so very long pages might be slightly slow to render or export as PDF [15], but generally it's fine for long-form text. BookStack encourages structuring content into chapters, which helps avoid single overly-large pages. It also generates an on-page table of contents for long pages automatically.	Nextcloud stores pages as files (Markdown), so it can handle a lot of notes. Many pages are fine, and moderate page sizes are fine. A *very* large single page (e.g. dozens of MBs of text/images) could be taxing on the browser when editing, but normal documentation sizes are supported. In practice, users can create multiple pages in a Collective to split content. There is no known hard limit on notebook size; Nextcloud's file backend can handle large data volumes.

| **Collaborative Editing** | **No (not real-time):** DokuWiki does **not** support concurrent editing on the same page. It uses a basic lock mechanism – when one user is editing, others are prevented to avoid conflicts. Changes must be saved to be seen by others. No live cursor or simultaneous edits (this limitation is typical of lightweight wiki engines) [1] . *(Plugins for real-time collab are not common, so assume single-user editing at a time.)* | **Yes:** Confluence has a **real-time collaborative editor**. Up to 12 people can edit the same page simultaneously, seeing each other's changes live [2] [16] . This "shared draft" mode syncs changes automatically and greatly aids teamwork on meeting notes, specs, etc. (On Data Center, the Synchrony service enables this feature.) Collaborative editing is built-in and feels similar to Google Docs in practice [17] [18] . | **Yes:** Real-time collaboration is a core feature of Docmost. Multiple users can work on the same page concurrently without overwriting each other [19] . All contributors see updates in real time, making it very similar to Notion's multi-user editing. Docmost also supports comments on pages for asynchronous collaboration [20] . This is a major strength of Docmost (positioning it as a true collaborative wiki). | **No (no live co-editing):** BookStack currently uses a traditional edit model – one person edits a page at a time. There's no simultaneous multi-user editing; if a second user opens the editor, they might either be locked out or risk saving conflicts (BookStack will warn of edits made since opening the page). Real-time collaboration is **not** a feature (the developers note lack of live collab as a trade-off for simplicity [21]). Teams can collaborate by editing sequentially and using the page revision history, but not live together. | **Yes:** Nextcloud's Collectives app (and the underlying Text editor) allow **simultaneous editing** of markdown notes. Multiple people can edit the same page at once and see updates in real time [22] . This collaborative markdown editing is similar to Etherpad/Google Docs (it uses Operational Transform/CRDT under the hood). This means team members can co-author documentation pages concurrently. (Note: if using the simpler **Notes** app, that is essentially single-user notes or basic shared notes, whereas **Collectives** is meant for multi-user wiki pages with real-time editing.) | | **LDAP/Active Directory Integration** | **Yes (free):** DokuWiki has built-in LDAP authentication support via its plugins (the **authLDAP** plugin is bundled) [23] . This allows binding to Active Directory or other LDAP servers for user login. It's configured in the DokuWiki config (no extra cost – DokuWiki is OSS). Fine-grained ACL can then use AD groups if set up. (There's also a dedicated AD plugin, but the standard LDAP plugin often suffices.) | **Yes (paid, included):** Confluence Data Center supports direct integration with LDAP/AD for user directories. You can connect to AD, import users/groups, and use AD credentials to log in. This is included with the Confluence license (no separate plugin needed, it's a built-in feature of the enterprise edition). On Confluence Cloud, Atlassian Access would handle AD sync (with cost), but since we focus on on-prem, the Data Center license covers LDAP support. | **Yes, but *Enterprise only*:** Docmost's free open-source edition does **not** include LDAP support. LDAP authentication is listed as an **Enterprise Edition** feature [24] . (Similarly SSO/SAML and TOTP 2FA are enterprise features [25] .) This means that out-of-the-box, the community edition uses its own user management (email/password accounts). For AD integration, a paid license is needed [26] . This is a key consideration if centralized auth is required without budget for enterprise licenses. | **Yes (free):** BookStack has **built-in** LDAP/AD authentication support. Admins can configure LDAP details in the .env or config, and users can authenticate against AD servers [27] [28] . It's completely free and included in the open-source project. BookStack also supports OAuth2/OIDC and social logins (including Azure AD via OIDC) out-of-the-box [13] . No paid edition — all features are available under its open-source license. | **Yes (free):** Nextcloud has **robust LDAP/AD integration** built-in. By enabling the **LDAP user and group backend** app, an admin can bind Nextcloud to an AD/LDAP server and have users appear and authenticate with AD credentials [29] [30] . It supports group import, login attributes, avatar sync, etc. This is a standard feature in Nextcloud's community edition and widely used in enterprises. (In short, Nextcloud will seamlessly connect to AD — *"seamless connectivity to Active Directory, with no extra configuration required"* [30] .) | | **Attachment Support (Files & Images)** | **Yes:** DokuWiki allows file attachments (images or documents) through its **Media Manager**. Users can upload files (e.g. PDFs, Word, images) to the wiki's media folders and then embed or link them on pages [31] . By

default, common file types (like `.pdf`) are allowed for upload [31]. When editing, clicking the "Add image/file" button opens the media manager to upload and insert files. Attachments aren't displayed inline (except images), but appear as download links on the page. DokuWiki doesn't index attachment contents for search (only file names). Overall, it has basic but adequate attachment support for reference files. | **Yes (robust):** Confluence has first-class attachment support. You can **drag-and-drop** files onto a page or use the *"Insert files/images"* toolbar button [32]. Files (PDFs, Office docs, images, etc.) get attached to the page and can be displayed as links or as preview thumbnails/embedded content. Confluence *indexes attachment content* for search – e.g. PDFs and Word docs are text-extracted so you can find text inside them [33]. It also provides versioning for attachments (uploading a file with the same name creates a new version) [34]. Additionally, Confluence can generate previews: for example, an attached PDF or PowerPoint can be viewed within Confluence via the file previewer (or using the PDF macro) [35]. These features make attachments very convenient for documentation repositories. | **Yes:** Docmost lets you attach files to pages for easy reference [36]. There is an "Attach file" option on pages. Like Confluence, attached files are listed and can be downloaded by readers. (Enterprise edition even supports searching within attachments like PDFs/DOCX [36].) Images can be embedded inline in pages. From user feedback, one limitation is that currently attachments persist even if not used (no global file manager to remove unused files) [37]. But functionally, you can upload screenshots, PDFs, etc., and either embed them or link them in the documentation. | **Yes:** BookStack pages can have files and images attached. In the editor, there's an **Attachments** sidebar where you can upload files (or attach by drag-and-drop) [38]. Images can be inserted into page content, and other file types (PDFs, spreadsheets, etc.) can be uploaded and then linked within the text or simply listed as attachments. BookStack manages attachments per page: the files are stored and shown in a list on that page's sidebar [39]. Users with view access to the page can download its attachments [40]. (BookStack does not index attachment contents for search, as it expects page text to hold the documentation [41].) Overall, attaching screenshots or documents in BookStack is straightforward, and you can even **drag attachments into the editor** to create links automatically [42]. | **Yes:** Nextcloud's platform is fundamentally a file storage system, so it handles attachments well, albeit a bit differently. In **Collectives**, you can embed images in pages (Nextcloud 24+ allows direct image uploads in the text editor [9]). For other files (PDFs, etc.), typically you would upload the file to Nextcloud (perhaps in a folder) and then link to it from the Collective page. There is a *"link file"* option to easily link local Nextcloud files into a page [22]. This way, you can provide downloads or reference documents. One caveat: earlier versions required manual steps (pasting images from clipboard didn't work initially [43], but improvements were made to allow uploading images directly). By now, adding attachments is supported via the UI (e.g. an "Add file" button to pick a file from Nextcloud or upload new). While attachments aren't listed at the bottom of a page like in Confluence/BookStack, they are accessible via the links you insert. And since Nextcloud has built-in PDF viewers and Office integration (Collabora/OnlyOffice), users can click attached file links to view them in browser. | | **Export / Import Capabilities** | **Limited (plugins for more):** DokuWiki pages are stored as plain text files, so one can always retrieve the raw .txt. By default, DokuWiki offers an **HTML view** of pages for printing, and it can generate RSS or use tools to produce other formats [44]. There is no one-click PDF export in core; however, popular plugins exist (e.g. **DW2PDF** plugin) to export a page or a collection of pages to PDF [45]. It also has an ODT export plugin (for OpenDocument text). For HTML export: you can use the built-in "Site Export" feature to dump all pages to HTML (and even create a static site). So, while DokuWiki can output content, it may require installing plugins or using manual methods. Importing content: DokuWiki doesn't have a native importer for other formats, but community scripts exist (for example, to convert HTML or Word docs to DokuWiki format [46]). Migration into DokuWiki often means converting files to its wiki syntax or simply copy-pasting and cleaning up formatting. | **Rich Export/Import:** Confluence supports multiple export formats out-of-the-box. Individual pages can be exported to **PDF** or **Word** with a couple of clicks [47] [48]. Entire spaces (or selected sets of pages) can be exported to PDF (one PDF containing all pages) or to a zip of

HTML pages [49] [50]. An XML export is also available for backup or migration purposes [51]. These exports preserve formatting, images, and attachments (attachments can be included in HTML/XML exports) [52]. Confluence's PDF exports are quite polished – you can even customize the PDF styling (cover page, headers/footers, etc.) if needed. **Import:** Confluence can import content from Word documents (it will convert a .docx into a Confluence page). There are also marketplace tools or built-in support for importing Confluence space XML (for migration between instances) and even an importer for Microsoft Word and some markdown import options. (No direct OneNote importer exists, but one could export OneNote pages to Word or PDF then import.) Overall, Confluence's export/import options are strong and suitable for enterprise needs [53]. | **Import/Export present:** Docmost can **import and export** pages in **Markdown or HTML** format [54]. This means you can export a page (or an entire space) as Markdown files or HTML files, which is great for portability. It even has importers: notably, it can import a collection of Markdown/HTML (zipped) and has specialized import for Notion pages, and (in Enterprise) a Confluence importer [55]. PDF export is not explicitly mentioned in documentation – presumably one would use the browser's print-to-PDF if needed (there's no one-click PDF generation in the community version as of now). For most documentation, Markdown/HTML covers use cases (you could convert markdown to PDF via external tools if necessary). In summary, Docmost allows getting your data out in open formats, which is useful for avoiding lock-in. *(Note: Since it's newer, these features are evolving; always check the latest docs for additional export formats.)* | **Comprehensive Export:** BookStack provides built-in export options for pages, chapters, or whole books. You can export content as **HTML** (a single HTML file per page, with embedded images), as **PDF** (auto-generated PDF of the page/book), as **Markdown** (`.md` file), or as plain text [56] [57]. For example, a page or an entire book can be exported to PDF with a click (it uses an internal PDF generator) [58]. These export features make it easy to create offline copies or share documentation outside the system. **Import:** BookStack supports **importing** content via a **"Portable ZIP"** format [59] [60] – essentially a zip containing HTML/Markdown and attachments in a specific structure (for moving data between BookStack instances or from other sources if formatted correctly). It does not natively import Word or OneNote directly, but you could convert those to Markdown/HTML and then import. There is also an API that could be used for custom migrations. Overall, BookStack's export/import is quite good for an open-source tool, supporting multiple formats including easy PDF exports [56]. | **Basic Export (via browser or scripts):** Nextcloud Collectives doesn't have a dedicated one-click PDF/HTML exporter for pages as of now. However, because pages are in Markdown, there are workarounds: there is a "Print" option which effectively allows you to print or save a page as PDF through the browser [61]. (The Collectives FAQ notes that the *Print* function was provided as a shortcut to PDF export via browser [62].) Also, since each page is stored as a `.md` file on the server, an admin could directly retrieve Markdown files or use a tool (e.g. a Nextcloud plugin or external script) to batch convert them. Nextcloud also introduced a **"External markdown editor (Text)"** which from Nextcloud 31+ with a Pandoc integration can save Markdown files as PDF or DOCX via a workflow [63]. In short, out-of-the-box you'd use print-to-PDF or manually grab the markdown. There's no native multi-page export or whole-notebook export yet. Importing content would similarly be manual (copy-paste Markdown or drop pre-written `.md` files into the Collectives folder structure). Nextcloud's focus is less on fancy export and more on live collaboration, but improvements are ongoing (e.g. community scripts for markdown conversion [64]). |

| **Mobile Access** (FYI) | **Partial:** No official mobile app for DokuWiki, but the web UI is lightweight and can be accessed via mobile browser (though not touch-optimized). Some responsive themes exist to improve mobile view. Mobile editing is possible in a pinch, but it's not a great experience for heavy use. | **Yes:** Confluence has a mobile app and a responsive web UI. The Confluence Server/Data Center mobile app allows viewing and basic editing of pages on the go (though advanced editing can be tricky on mobile). For our scope, mobile isn't required, but know that Confluence provides good mobile support if needed. |

Limited: Docmost does not yet have a dedicated mobile app. The web interface is responsive, so users *can* read pages on mobile browsers. Editing on mobile might be less optimal given the rich editor. (As Docmost is new, mobile support may improve, but currently it's primarily desktop/web focused.) | **Yes:** No mobile app, but BookStack's UI is responsive. Users can view documentation on phones/tablets via browser reasonably well. Editing on a small screen is possible (the editor works in mobile browsers), but large documentation edits are more comfortable on desktop. Since mobile access is not a priority here, this may not be a deciding factor. | **Yes:** Nextcloud has mobile apps, and while those are mainly for file access, the Collectives app pages can be accessed via the Nextcloud mobile app's interface (or a browser). The Nextcloud *Deck* app added Collectives integration recently, and the *Nextcloud Notes* app on Android can sync simple notes (not full Collectives though). In essence, one *could* read and lightly edit notes on mobile browser or via some app, but it's not as seamless as OneNote mobile. (Mobile not required for the team, so this might not matter.) | | **On-Premise Deployment** | **Excellent:** DokuWiki is very easy to self-host. It's just PHP and flat files (no database), so you can deploy on any LAMP stack or even a shared hosting. There's an official Docker image available [65] for quick setup, and it's light on resources. It will run comfortably in a small VM or container. Fits well in both VMware VM or Kubernetes (a simple container can hold it, and no external DB simplifies deployment). Backup is simple (file copy). Upgrades are manual but straightforward. | **Yes (with costs):** Confluence can be run on-prem on a VM or Kubernetes, but requires a **Data Center** license. Atlassian provides Linux installer packages and also Docker images for Data Center. It's heavier: requires Java and typically a database (PostgreSQL, etc.) plus an external shared storage if clustered. Kubernetes deployment is possible (Atlassian offers Helm charts for Data Center). In VMware, you'd likely run it on a dedicated VM (with enough CPU/RAM). Note that Confluence is resource-intensive compared to the others and the licensing cost for Data Center is significant, especially beyond 10 users [66]. However, it is fully on-premise capable and often used in enterprise VM environments. | **Yes:** Docmost was designed for on-prem. It requires a Node.js runtime, PostgreSQL, and Redis, but the developers provide a **Docker-based** installation which simplifies deploying all components [67]. You can run that Docker image on a VM or orchestrate it in Kubernetes (it might not have an official Helm chart yet, but community could adapt the Docker). Technical skill is needed to set up the environment variables and persistence (as with any custom web app). In summary, it's deployable on-prem, but **more complex to set up** than PHP apps due to its modern stack [68] [5]. Ensure your team has the know-how to manage a Node/Postgres app in production (and be mindful that enterprise features require license keys). | **Yes:** BookStack is an open-source Laravel (PHP) application with MySQL/MariaDB. It's quite easy to host on a VM or container. There's an official Docker image and a well-documented install script for Ubuntu, etc. It runs on a standard LAMP/LNMP stack (PHP 8 & MySQL) [69]. Many users deploy it via Docker Compose or on a small VM. It's lightweight (~<100MB RAM usage typically) and fits well even in modest infrastructure. Kubernetes deployment is feasible (several community Helm charts exist). Thus, BookStack scores high for self-hosting ease and low maintenance overhead. | **Yes:** Nextcloud is **built for on-premises** by design [70]. It can be installed on a VM (LAMP stack) or via Docker (official images) and there are community Helm charts for Kubernetes. Nextcloud is heavier than a simple wiki (because it includes many apps and requires PHP, a database, plus storage for files), but it's well-documented and widely deployed self-hosted. In VMware, one might use a VM with the Nextcloud VM image or install from scratch. In K8s, you'd run Nextcloud with a database and maybe use something like an NFS or S3 for storage. Since Nextcloud likely may already be in use for file sharing in some organizations, adding the Collectives app is just an extra step. In sum, hosting Nextcloud on-prem is very common – you maintain full control and can leverage enterprise features with the free community edition. |

Detailed Evaluation by Criterion

1. Ease of Use for Long-Form Documentation

DokuWiki: Suitable for tech-savvy users who don't mind wiki syntax. It supports all basic formatting (headings, bullet lists, numbered lists, tables, images) through its plain-text syntax. For instance, creating a table or linking an image requires using DokuWiki markup (which some non-technical users may find unintuitive at first). The interface is spartan and a bit dated – essentially a page view with "Edit" button, which opens a textbox for editing. However, a toolbar in the editor provides buttons to insert common syntax (bold, italics, link, etc.). Non-technical staff **can** learn it, but there may be a learning curve. A WYSIWYG editor is *not* in core; if user-friendliness is a major concern, an admin could install a WYSIWYG plugin (e.g. the ProseMirror or CKGEdit plugin) to allow MS Word-style editing – but these plugins vary in quality. Navigation in DokuWiki is also less visual: by default it doesn't show a sidebar page tree (content is organized by "namespaces" which act like folders). Users might have to rely on search or an index to find pages, unless you add a sidebar manually or use a plugin for a navigation menu. In summary, DokuWiki's strength is simplicity and reliability, but its UI and editing experience are not as slick as the others. It's often described as *"focused on function over form"* – great for those who just need to write and don't mind text markup [1] [10], but perhaps intimidating to OneNote users accustomed to a rich interface.

Confluence: Very user-friendly for long-form docs. Its editor allows rich text editing (bold, fonts, colors, headings), and inserting images or tables is point-and-click. The experience is similar to writing in a word processor or OneNote – you can paste screenshots directly and they appear inline [32] [71], and create tables with a GUI (adding/removing rows, merging cells, etc., all with buttons). Confluence handles large pages well, automatically generating a table of contents if you use the TOC macro. The UI is intuitive: a sidebar shows the page hierarchy in a space, so users can see the structure of documentation. For non-technical staff, Confluence is often the easiest to pick up, because it doesn't expose them to any markup at all. The editing toolbar and macros (like info panels, status lozenges, etc.) make it easy to add useful formatting without technical knowledge. Confluence's interface is modern and polished, often described as "it just works" for creating neat documentation. The downside is that Confluence might encourage **complex** pages with many macros; but for basic note-taking and documentation, it excels in ease of use. OneNote users will find Confluence familiar in that you can click and type (though *not* anywhere absolutely free-form; Confluence is still a structured document editor, not an infinite canvas). But it does allow some layout freedom with column macros (you can create a 2-column section, for example, to put content side by side). Overall, Confluence offers the most refined editing experience among these, akin to a professional documentation tool.

Docmost: Docmost was designed as an open-source alternative to Notion/Confluence, so usability is a strong focus. The interface is **modern and clean** [5]. The editor supports Markdown shortcuts – meaning you can type in a lightweight way (e.g. start a line with "#" for a heading, or "-" for a list) and it will format accordingly, but you also have rich text controls so you don't need to know Markdown if you don't want to. This makes it friendly to both technical and non-technical users. Adding images or attachments is straightforward (there are toolbar options or drag-and-drop). Docmost includes nice touches for documentation: it natively supports embedding diagrams (via Draw.io, etc.) which is great for technical docs [4], and it allows page nesting with an automatic sidebar navigation in each Space. Non-technical staff should find the experience comparable to tools like Notion – a clean workspace with blocks of content you can edit easily. The UI also likely has on-page tips or minimal clutter, making it pretty intuitive. Since Docmost is relatively new, its interface is modern web design – for example, it might use a block-based editor where each paragraph or image is a block you can move or comment on. OneNote users will appreciate the ability to just start writing without worrying about wiki syntax. However, unlike OneNote's absolute freeform canvas, Docmost pages are still linear documents (which is true of all these wiki tools). That said, the learning curve for Docmost should be low, and its **real-time nature** means users see changes as they type, reinforcing ease of use.

BookStack: BookStack's UI is often praised for its simplicity and intuitiveness, even for non-technical teams. The bookshelf metaphor (Shelves -> Books -> Chapters -> Pages) provides a familiar mental model – it feels like organizing content in a digital binder with tabs. Creating and editing pages is done through a WYSIWYG editor by default, which is very much like typing in Word or OneNote. Users can click buttons to insert headings, format text, create lists, tables, hyperlinks, etc., without needing any syntax knowledge [6]. There is also a Markdown editor option (with preview) for those who prefer or for certain content types, but typical users will stick to the WYSIWYG. Screenshots and images can be added easily (drag-and-drop into the editor or using the image upload button). The integration with diagrams.net means even non-technical users can add flowcharts or diagrams by clicking an "Insert Drawing" button and using a visual editor [72] – a big plus for replacing OneNote drawings or Visio diagrams. BookStack's page editor also has an **Attachments sidebar** for adding files, which keeps the interface uncluttered but powerful when needed [38]. The platform emphasizes a **clean reading experience** too: when viewing pages, the UI shows content clearly with a navigation pane on the left. One potential adjustment for OneNote users: BookStack enforces a structured hierarchy (every page belongs to a chapter & book, except standalone pages in a book). Users who are used to a more ad-hoc structure might need a little planning to organize content logically into books and chapters – but this structure can be beneficial for consistency. Overall, BookStack is very friendly; most users "become familiar with it in no time" [73] [74], and it's specifically aimed at being easy and appealing for employees.

Nextcloud (Notes/Collectives): Nextcloud's Collectives app provides a **basic but effective** interface for documentation. It is essentially a simple Markdown editor embedded in Nextcloud. This means the editing experience is not as feature-rich as Confluence or BookStack, but it covers the essentials: you type content (the editor will render bold, italics, headings, etc. as you type markdown). It's described as a *"lightweight, distraction-free text editor that lets multiple users edit text with basic needs like bold, headers, bullet points and images"* [75] [8]. The interface shows the content in a clean way – for example, headings are larger, bold text appears bold, etc., so users aren't stuck looking at raw markup. In Collectives, on the left you have a list of pages, and on the right the content of the selected page (very similar to a Notes app). There's no complex menu system; just a top bar with a few icons (edit, share, etc.). For non-technical staff, using Collectives should be quite straightforward for writing text. However, some **limitations** in formatting exist: advanced styling (different fonts, colors, highlighting) is not really available – the philosophy is that if *"bold, headers, bullet points and images cover most of your needs, Nextcloud has you covered"* [75]. Inserting tables was initially missing but is supported in Nextcloud 24+ (you can create tables in markdown and they will display properly [76]). OneNote power users who are used to freely positioning text boxes or mixing drawings within text might find Collectives *too plain* – content is strictly linear and formatted by markdown rules. Also, Nextcloud doesn't provide a dynamic table of contents for a page or sophisticated templates (each page is a blank document). On the plus side, the simplicity means almost no learning curve: if you can write an email, you can write in Collectives. And the integration with Nextcloud means if users are already familiar with Nextcloud's UI, this is just another app in the menu. In summary, Nextcloud Collectives is easy for basic note-style documentation, but it might feel *less powerful* in formatting options compared to a dedicated wiki like Confluence or BookStack. It's best suited if your formatting needs are simple and you value integration over fancy UI.

2. Collaborative Editing

DokuWiki: It does **not support real-time collaborative editing**. DokuWiki's model is that one person edits a page at a time. When you click "Edit", DokuWiki can lock the page (by default, it will warn others that "User X is editing, lock acquired") to prevent edit conflicts. If someone else tries to edit simultaneously, they'll either be prevented or risk an edit conflict upon saving. There is no live merging of changes. Essentially, collaboration in DokuWiki is sequential: you make your edits, save, then someone else can edit. This is a significant difference from OneNote (which on OneDrive can merge edits concurrently) or Google Docs style editing. Users used to seeing each other type in real-time will **not** get that in DokuWiki. There might be a plugin or workaround for basic concurrent editing, but it's not mainstream. The lack of real-time editing is acknowledged as a limitation for modern collaboration [1] [77]. However, DokuWiki does track revisions – each save is a new version, and you can compare differences and revert if needed. So multiple people can collaborate by editing one after the other and relying on the history to track who changed what. In practice, for teams that are not editing the *same* page at the exact same moment, this might be sufficient. But if your workflow involves people simultaneously editing meeting notes or brainstorming on a page, DokuWiki will not be ideal.

Confluence: Excellent collaborative editing. Confluence supports real-time multi-user editing on a page (in Data Center since version 6.x). When multiple people click "Edit" on a page, they will enter a shared editing session. You can see indicators of who is currently in the page with you, and their cursor or the text they're modifying updates live on your screen [17] [78]. It allows up to **12 concurrent editors** by default [79], which is usually more than enough. Everyone's changes are saved to a shared draft continuously (and automatically), so you don't have to worry about overwriting each other [80] [81]. This is great for teamwork – for example, during a meeting multiple people can contribute to notes on the same Confluence page in real time. There are a few caveats: if many people try to edit a very large page, it can get a bit slow or occasionally glitch, but those cases are rare. Also, until changes are published, they exist as an "unpublished draft" visible to editors but not others, which Confluence handles behind the scenes. Confluence's approach is quite mature – it's analogous to Google Docs (they use an Atlassian-developed service called Synchrony to merge edits). This feature really "takes teamwork to the next level" by letting the team capture information together live [17]. If your users are accustomed to OneNote multi-user editing (which OneNote does via cloud sync), Confluence will meet that need and perhaps be even smoother in some cases. Overall, Confluence is the **strongest** in collaboration capabilities, tied closely with Docmost in this comparison.

Docmost: Real-time collaboration is a core strength. As an open-source alternative to Notion/ Confluence, Docmost has invested in true concurrent editing. Multiple users can open the same page and edit concurrently, and everyone sees changes in real time [19]. This means no page locking – you can collaboratively create or update content, which is crucial for a dynamic documentation team. It likely shows user cursors or at least immediate updates of text. Being a modern application (built on frameworks that support websockets or similar), the experience should be fluid. Additionally, Docmost has features like inline comments on pages [20] to facilitate discussion, which complements the editing experience (not exactly multi-user editing but a related collab feature). Essentially, Docmost aims to match Notion's level of collaboration – in Notion, any number of people can edit a workspace simultaneously. Docmost's docs even emphasize *"seamless real-time collaboration, multiple users can work on the same page without overwriting"* [19]. So it's safe to say it delivers true concurrent editing. This is a major plus for teams that often collectively edit knowledge base articles or specification documents. One should note that because Docmost is newer, you might occasionally find small quirks (like the ordering of overlapping edits or needing a stable connection), but generally it's praised and "most users praise its functionality" in this area

[82] [5]. So for collaborative editing, Docmost is on par with Confluence in capability – outshining DokuWiki and BookStack which have none.

BookStack: No real-time concurrent editing. BookStack uses a simpler approach: one person edits at a time. If a second person clicks edit, they will be editing the last saved version of the page – there isn't an internal mechanism to merge changes from two live editors. BookStack does have a feature where if you open a page to edit and someone else saves changes before you save, it will detect a version mismatch. In such case, I believe BookStack might show an error or a warning when you attempt to save, to prevent silent overwrites. But effectively, simultaneous editing is not supported. BookStack's maintainers have explicitly noted that *"it does not provide real-time collaboration"* [21] – it's a trade-off chosen to keep the app lightweight and simple. Teams using BookStack usually coordinate their edits (perhaps by working on separate pages or sections) to avoid stepping on each other's toes. The page revision history is available, so if someone overwrote someone else's change by mistake, one could retrieve the older version manually. For many documentation scenarios (like writing manuals or how-to guides), this lack of real-time editing is not a deal-breaker – people often work independently on different topics. But if your team is used to co-editing the *same* page actively (like co-authoring meeting minutes), BookStack will feel limiting. Each user will have to wait their turn or split the notes and later compile them. In short, collaboration in BookStack is asynchronous – good version control but no simultaneous edits.

Nextcloud Collectives: Yes, it supports live collaboration. Using Nextcloud's *Text* editor, Collectives enables multiple people to edit a page together with changes merging in real time [22]. This was a big feature introduction in Nextcloud (branded *Nextcloud Text*), touted as *"an easy to use, collaborative rich-text editor"* where *"any number of people [can work] at the same time, sharing ideas in real time."* [83]. Practically, if two users open the same Collective page, they each see what the other types almost immediately. This is powered by the underlying technology (likely using WebSockets and conflict-free data types). The collaborative editing in Collectives is not as graphically elaborate as Confluence's (for instance, you might not see colored cursors with names), but functionally it's similar to Google Docs' simultaneous editing on a plain text document. There is also no user limit documented; presumably it can handle a good number of concurrent editors (Nextcloud demonstration cases often show 5-10 people editing a text file together). So for capturing notes in a meeting or collectively building documentation outlines, Nextcloud works well. One difference: because the editor is Markdown-based, if two users are, say, both creating different list items or headings, they need to be mindful of not breaking the markdown syntax – but in practice it's not difficult, and the system merges text changes line by line. To summarize, Nextcloud provides *true concurrent editing*, making it a viable choice if real-time collaboration is needed. This sets it apart from older wiki platforms and puts it closer to Confluence/Docmost in this regard. For a team coming from OneNote: OneNote (when synced via Office 365) also allows multi-user editing, and Nextcloud can replicate that experience of multiple people adding content to a page at once (with the advantage that changes are saved instantly to the server for all to see).

3. LDAP Integration (Authentication)

DokuWiki: Supported out-of-the-box. DokuWiki has been used in many corporate environments and includes an **LDAP auth plugin** with the core distribution [23]. Configuring DokuWiki to use LDAP/Active Directory involves editing its config file (or using the Configuration Manager in the admin panel) to set the LDAP server address, bind DN, search filter, etc. Once configured, users can log in with their AD credentials instead of separate DokuWiki accounts. Group memberships can also be synced (for example, you can map an AD group to a DokuWiki group for ACL purposes). The DokuWiki site provides example configs for AD

[84], and generally it’s considered straightforward. Importantly, this doesn’t require any paid plugin or enterprise version – it’s part of the free software. So, in terms of **cost**, DokuWiki is great: you get LDAP integration for free. In terms of **ease**, it might require a bit of manual setup and ensuring the PHP LDAP module is enabled on the server. But it’s a one-time configuration. Once done, DokuWiki will seamlessly authenticate users against your domain, meaning no additional user management hassle. This satisfies the requirement for central auth integration.

Confluence: Supported (in Data Center license). Confluence (Server/Data Center) allows connecting to external user directories including LDAP and Active Directory. In the Confluence admin UI, you can set up an LDAP directory, specifying the LDAP URL, credentials, base DN, etc., and Confluence can either sync users or do on-the-fly LDAP authentication. This is not an extra cost beyond owning Confluence – it’s a built-in feature Atlassian provides, since enterprise customers commonly use AD. For instance, Confluence Data Center supports Microsoft Active Directory, OpenLDAP, and others; it can even pull in group membership for permission assignments. The **caveat** is that Confluence itself is a paid product (free for 10 users in old server edition, but now effectively one needs a Data Center subscription which starts at 50 users). So while technically there’s no “enterprise addon” fee for LDAP, you are paying for the product itself. Assuming you have that, integrating with AD is usually straightforward and Atlassian has documentation to guide it. In summary, Confluence will **check the box for LDAP support** easily – you’ll have unified login for your users (or potentially single sign-on via something like SAML if needed, though SAML SSO might require an additional configuration or the use of Atlassian Access for cloud). Since the question specifically asks if it’s free or paid: with Confluence, **you must pay for the software**, but *within* that, the LDAP integration feature is included (no separate plugin purchase needed).

Docmost: LDAP is only in the paid Enterprise version. According to Docmost’s official docs, LDAP authentication is **not available** in the open-source AGPL edition; it’s part of the “Enterprise Edition” features [25]. This means if you deploy the free/community Docmost, you only have email/password auth (and possibly OAuth with some common services like Google, if they implemented that – though that might also be enterprise). The enterprise license unlocks LDAP and SSO (SAML/OIDC) [25] [26]. This is an important consideration: if integrating with AD is a must and you want Docmost, you should budget for its enterprise plan or be prepared to write an auth integration yourself (not trivial). The Enterprise Edition likely comes with support as well, but that’s an extra cost (the Docmost site says “contact sales” for enterprise licensing). The user count and pricing aren’t publicly listed, so you’d have to reach out. Depending on budget, this could still be cheaper than Confluence for a small team, but it is a negative against Docmost in a purely free/open-source comparison. If you forego LDAP, you’d manage users within Docmost itself, which is a disadvantage in an enterprise setting (duplicate accounts, manual provisioning). So, to answer clearly: **LDAP integration in Docmost is paid-only** [24]. If the team cannot purchase, this might eliminate Docmost or require a workaround (like using an OAuth against an AD-federated service if available).

BookStack: LDAP/AD supported natively (free). BookStack includes built-in support for LDAP authentication configuration. An admin can open the BookStack `.env` file (or use the environment variables) to set `AUTH_METHOD=ldap` and provide the LDAP server, base DN, user filter, etc. BookStack uses PHP’s LDAP functions to authenticate users. There are detailed docs and community examples for Active Directory specifically [27] [28]. It even supports syncing LDAP groups to BookStack roles (so you can manage permissions via AD groups) [85]. Crucially, this functionality is part of the core open-source project – no fees or editions. So you can integrate with AD at no cost besides your time setting it up. Many BookStack users on forums have successfully connected to Windows AD for login, which shows it’s a well-trodden path (make sure the PHP LDAP extension is installed on the server, as noted in docs [86]). One nice aspect: since

BookStack is self-hosted and open, if you needed to customize anything about the LDAP integration, you could. But generally it covers common needs (multiple LDAP server support, filters, etc.). Therefore, BookStack gets full marks for LDAP integration: it's **free and fully functional** for AD logins.

Nextcloud: Full LDAP integration included (free). Nextcloud is known for its strong focus on integration into enterprise environments, and LDAP/AD is one of its core features. By enabling the LDAP app in Nextcloud, you can connect to one or multiple LDAP servers [29]. The Nextcloud admin interface provides a wizard with multiple tabs (Server, Users, Groups, etc.) to configure the connection and filters. It supports a wide range of LDAP configurations (Active Directory, OpenLDAP, FreeIPA, etc.), including login attribute mapping, group membership, and even avatar photo import from AD [87] [30]. Nextcloud's LDAP integration is quite seamless – users will appear in Nextcloud's user list as if they were local, and you can assign them to Nextcloud groups or give them quotas, etc., while their password and authentication are deferred to the LDAP server [87]. It's also worth noting Nextcloud can integrate with AD's LDAP **and** also support single sign-on systems (like SAML) via free apps, but focusing on LDAP: it's included in the community edition and widely used. No extra license is needed (unless you opt for **Nextcloud Enterprise** for official support, but the feature itself is in the free version). Many organizations use Nextcloud as a front-end to their AD-authenticated user base [30]. So, Nextcloud easily meets the LDAP integration criterion. In fact, out of all options, BookStack and Nextcloud are the ones where LDAP support is both free and well-developed, while Confluence is well-developed but not free, and Docmost requires a paid upgrade.

4. Attachment Support

DokuWiki: Attachments are handled via the **Media Manager**. DokuWiki doesn't use the term "attachment" in the same way Confluence or BookStack do (with a list per page), rather any file (image or document) you upload is stored in a namespace in the `/data/media` directory. When editing a page, there's an "Add Images and other files" button that opens the media manager popup [88]. There you can choose a file to upload (and select which namespace folder to put it in, often same as the page's namespace). Once uploaded, you insert a link or image syntax into the page. For example, to attach `document.pdf` and link it, you'd upload it and then DokuWiki inserts something like `[[document.pdf]]` in the page which appears as a link. For images, it would insert `{{image.png}}` which displays the image. By default, DokuWiki allows most common file types (the config `mime.conf` controls allowed extensions; PDF is allowed by default) [31]. Users can download attached files by clicking the links. There's no preview for PDFs or docs, it's just a download. Attachments are protected by the same ACL as the pages – if you can read the page, you can fetch the file. One limitation is **organization**: attachments are not listed automatically on the page. You either manually make a list of links or rely on a plugin that shows media files in a namespace. Also, if you delete a link from a page, the file stays on disk (unused files aren't tracked unless using plugins). Searching in DokuWiki does not search file contents (only page text). So attachments in DokuWiki are more bare-bones: fine for providing downloadable files or embedding images, but not as integrated as in some other tools. Still, it covers the requirement of being able to upload and attach PDFs or other documents to documentation pages.

Confluence: Confluence has **excellent attachment features**, treating attachments as first-class objects on a page. Users can simply drop a file into the editor or use the "Files & images" toolbar button to upload [32]. Once uploaded, files are listed under the page's *Attachments* (there's an Attachments view per page accessible via the tools menu) and can be versioned [34]. For usage in content: if it's an image and you drag it in, it gets embedded inline. If it's a PDF or other doc, Confluence by default will show it as a filename link, but you can also use macros like *View File* or *PDF* macro to display a preview or embed a PDF viewer.

Confluence's ability to index attachments for search is a standout feature – e.g. if you attach a PDF containing the text "XYZ", a user searching Confluence for "XYZ" can find that page because Confluence extracts the text from attachments and indexes it [33]. This is great for a documentation-heavy environment where some content might only exist inside an attached PDF or Word spec. Confluence also allows in-browser viewing for certain file types: images show a preview on click, PDFs can be read in a scrolling preview, Office docs can be viewed or edited with an Office integration. This reduces the need to download files just to see what's in them. OneNote users who used to embed printouts of documents will find that Confluence's approach (link + preview) is effective, though not exactly the same as having the document pages visible inline (except you can manually embed each page of a PDF as an image if needed, but usually linking is fine). There's also no hard size limit except what admin sets (default 100 MB, configurable) [89]. In short, Confluence is top-tier for attachment support, making it easy to include supporting documents in your wiki.

Docmost: Docmost supports attachments on pages – an essential feature for a documentation platform. According to its feature list, you can *"attach files to your pages for easy reference and sharing."* [36]. So functionally, you can upload files (screenshots, PDFs, etc.) to a page. Users can likely do this via an "Attach file" button or by dragging into the editor (assuming the editor supports that). Once attached, files might show up as small icons or a list below the page content. Docmost Enterprise adds the capability to **search within** PDF and DOCX attachments [36], which is similar to Confluence's indexing, but note that's enterprise only. In the free version, attachments are still usable, just their content isn't indexed. Embedding images works – Docmost's editor is rich, so inserting an image will display it in-line. For other file types, you might get a download link. There have been mentions in community (like a YouTube review or forum) that Docmost currently doesn't have an obvious "media manager" – files attached stay in the system even if not referenced (potentially a cleanup issue) [37]. Also, there isn't a hierarchical file repository view; attachments are tied to pages (like Confluence/BookStack). But for end-users, attaching a file is easy and retrieving it is just clicking the link on the page. If migrating from OneNote, where people might have embedded documents in notes, in Docmost they would attach the document and perhaps rely on users to click to open it (similar to Confluence's approach). Overall, Docmost meets the requirements: you won't lose the ability to include supporting files alongside your documentation.

BookStack: BookStack handles attachments nicely. When editing a page, an **Attachments** panel is available that allows uploading files or linking to external URLs [90]. A user can drag a file from their computer into this panel or click "Upload File". Once uploaded, the file appears in the list for that page and you can insert a link to it in the page content with one click [91] [42]. Images can be inserted and will display; other files (PDFs, etc.) will show as a hyperlink (with the file name). After saving, when viewing the page, attachments are conveniently listed in the sidebar (with icons for file vs link) [39]. This makes it clear what files are attached to that page. Users can download them by clicking, or even preview if their browser supports it (for images it will open in browser; for PDFs, depending on setup, it might download or open in a new tab). BookStack doesn't parse the content of files, so search won't find text inside a PDF (similar to DokuWiki in that regard) [41]. Attachments are stored in the filesystem (or S3 if configured) and are deleted if the page is deleted (to avoid orphans) [92]. The system is designed for attachments to be supplemental – BookStack expects that **page text contains the documentation**, not the attachments, which are for supporting files [41]. But in practice, users can use attachments liberally for things like PDF outputs, forms, or images that don't need to be part of the text flow. BookStack's approach is very user-friendly: **permissions** for attachments inherit the page permissions (so if someone can read the page, they can download its attachments) [40]. Attaching screenshots or images works seamlessly – you can even paste images from clipboard when using the WYSIWYG editor (the image will be uploaded behind the scenes).

This is similar to OneNote’s ease of embedding screenshots. One difference: OneNote shows attachments (like PDFs) as icons within the note; BookStack will show them in a sidebar or as a link you inserted – either way, easily accessible. So BookStack fully satisfies the attachment requirement.

Nextcloud: Nextcloud’s entire architecture is file-centric, which influences how attachments are handled. In Nextcloud Collectives, an “attachment” would typically be a file stored somewhere in Nextcloud (either in the Collective’s folder or another shared folder) and then **linked** or embedded in the page. The Collectives app has a *“link local file”* feature where you select text and link it to a file from the Nextcloud storage [22]. This is actually a powerful concept – it means the file doesn’t just live in the wiki; it’s a file in your Nextcloud which can also be accessed through the Files interface, shared, synced to desktops, etc. If you insert an image via *“upload image”* while editing a page (a feature improved in NC 24) [9], likely that image file is saved in a special folder (perhaps a subfolder for that Collective or page) and then embedded. Other file types, you would create a link so that clicking it either downloads or opens that file (Nextcloud can open PDFs or Office docs in its viewer). One challenge observed: earlier versions of Collectives didn’t automatically share an attached file with other members of the Collective, causing *“image not visible to others”* issues if the image was uploaded to a private area [93]. But the FAQ indicates improvements: by Nextcloud 24, uploading images directly is supported and presumably they are stored in the Collective so all members can see them [9]. As of now, if you want to attach a PDF for example, you might upload it to a folder that the team has access to (maybe the Collective is tied to a Nextcloud Circle that manages access) [94]. Then from the page, link to that file. It will show as a clickable link (and possibly show a small preview if the Collectives app or Nextcloud Files generates one – not sure if Collectives shows a preview thumbnail for PDFs; there was a user asking for better previews [95]). In any case, Nextcloud definitely allows storing and linking attachments – it just doesn’t have a dedicated “attachments list” per page UI like others. But the integration with Nextcloud’s file handling is a plus: for instance, a user could update the attached file by just uploading a new version to Nextcloud (the link would then point to the updated file). And since Nextcloud indexes filenames and perhaps can index content with the Fulltext Search app (if configured with something like ElasticSearch), you might get some search on attachments, though not as straightforward as Confluence’s built-in indexing. In summary, Nextcloud meets the requirement of attachment support but in a slightly *indirect* way: it leverages the Nextcloud Files system. For a team used to OneNote: they can still attach docs, but they’ll need to understand that the file lives in the Nextcloud folder (OneNote used to embed files in the .one package itself). One advantage: Nextcloud makes it easy to later retrieve or manage all attachments because they’re files in the system, not hidden away.

5. Export Capabilities

DokuWiki: Without plugins, DokuWiki doesn’t have a fancy export button for PDF or Word. It relies on the fact that pages are plain text files and can be rendered as needed. The common way to export a page is to use the browser’s “Print to PDF” or copy-paste the content. However, the DokuWiki community has created solutions: e.g., the **dw2pdf** plugin can generate a PDF of the current page (or even a book of pages) with one click [45]. There’s also an **ODT export** plugin that can create OpenDocument text files, which then can be opened in Word [96]. In the default installation, users can click “Export: Raw” to get the raw wiki text, or append `?do=export_html` to get a plain HTML version of a page [44]. For whole-site export, DokuWiki has a command-line tool or plugins to dump all pages to HTML or to a Google-able static site. So, while *possible*, it’s not as user-friendly as something like BookStack’s built-in export menu. On the import side, DokuWiki doesn’t natively import other formats – you’d have to either copy content manually or use community scripts (for example, some have written scripts to convert HTML to DokuWiki syntax for bulk importing [46]). In context, if you have a lot of content in OneNote and you export it to HTML or text, you’d need to convert it to DokuWiki format to import properly. This likely involves some manual effort or custom scripting. So DokuWiki's score on easy export/import is lower – expect to rely on plugins or manual methods. It's workable (especially with the popular PDF export plugin that many DokuWiki sites use to produce PDFs for documentation), but not "one-click out of the box."

Confluence: Confluence is strong in export capabilities. You can **export an entire space** to a single PDF file (which will contain all pages in that space, nicely concatenated) [49] [50]. This is great for making a "documentation book" or manual from your wiki. You can also export to HTML – either one HTML file per page in a zip, or a single long HTML (the UI provides choices for normal vs custom export) [50] [97]. The HTML export is often used if someone wants to host the docs as a static site or needs an offline browseable version. Confluence also has **XML export** that is mainly for backup or migrating to another Confluence instance (or to import into Confluence Cloud, etc.) [51]. For individual pages, as mentioned, you have **Export to PDF** and **Export to Word** in the page menu [47] [48]. These are very useful for sharing a single page outside the system – for instance, exporting a How-To page to PDF to send to a client. The PDF and Word exports respect the published content (they won't include inline comments or drafts) [98] [99]. They also can be customized by an admin (e.g., adding company logo, custom header/footer in PDF). On the import side, Confluence can import Word documents (each Word doc becomes a page, preserving basic structure and images). There's also an HTML importer (if you have an HTML file, you can import it as a Confluence page, though sometimes formatting needs cleanup). Additionally, because Confluence is widely used, there are tools to migrate from other systems (Atlassian provides a CSV importer and some specific converters, and marketplace apps exist for things like Notion or MediaWiki to Confluence). For OneNote specifically, there's no official direct importer, but a workaround is to export OneNote to Word or PDF, then import or copy that into Confluence. In summary, Confluence's export options are **rich and user-friendly**, making it easy to get data out for various purposes [53]. It likely has the most robust export features in this comparison.

Docmost: Docmost's documentation mentions **Import/Export in Markdown and HTML** [54]. In practice, this means you can export your content as Markdown files, which is great for portability (Markdown is widely used and can be converted to other formats easily). Similarly, HTML export allows you to get a fully formatted version of pages. It also lists specific importers: Notion import (to help people migrate from Notion) and a Confluence import (the latter is enterprise-only) [55]. The presence of a Confluence importer in enterprise suggests Docmost can read Confluence space exports (probably the HTML or XML) and pull them in, which is handy if moving off Confluence. For exporting to PDF, as noted, Docmost doesn't explicitly say it can do PDF. Perhaps at this time it doesn't have a one-click PDF generator. However, since pages can be exported to HTML, one could take that HTML and convert to PDF via external tools if needed. Also, the web browser's print to PDF is always an option, albeit with not as polished output. Docmost focusing on Markdown is nice for integration with developer workflows (you could store the MD in git, etc.). It indicates the data isn't locked in a proprietary format. One might want to check if Docmost can export a whole space as a zip of Markdown files (that would be very useful for backup or migration). The documentation line suggests it has a Zip import, which likely pairs with a Zip export. The features list says "Zip import" [55] – maybe you can import a zip of MD files to quickly populate content. Possibly there's a corresponding zip export or at least you could collect the MD from an API or via each page. Since the question is about on-prem replacement, having Markdown export is certainly acceptable. Docmost also has an API (I suspect, since modern tools do) which could allow custom export if needed. So while not as polished as Confluence or BookStack in UI options for export, Docmost covers the basics and ensures you can get your info out in standard formats, avoiding vendor lock-in.

BookStack: BookStack shines in this area for an open source tool. Users with the right permission can export **pages, chapters, or entire books** in multiple formats: **PDF**, **HTML**, **Markdown**, and **Plain Text** [56] [57]. The HTML export is a single HTML file with embedded images (so it's a self-contained page) [100], which often gives the best fidelity to how it looked in BookStack. The PDF export uses an internal generator (dompdf) – it's decent for most content, though sometimes complex layouts might not be perfect [58]. Markdown export is very handy if you want to move content to another system or use it in a git repo; if the page was originally written in Markdown, it gives you that exact source, otherwise it attempts to convert from HTML to Markdown [101]. There's also a "Plain Text" export which strips formatting. For large-scale export, BookStack doesn't have a single-click for "everything" (like all books at once) through the UI, but you can do one book at a time. And as mentioned, there's an API which could script a whole export if needed. The *Portable ZIP* export is a special case: it packages the page(s) HTML plus images and metadata, allowing import into another BookStack instance [59]. It's essentially a backup of that content. Importing content: BookStack can import those portable ZIPs [60]. It doesn't natively import from Word or Confluence, etc., but one can convert those to Markdown/HTML and then import via the API or by placing them into a ZIP format BookStack expects. The community has also created some import scripts (e.g., there was a GitHub issue about OneNote to BookStack import, and at least one user wrote a script to convert OneNote's DOCX export to BookStack – although it's not officially supported) [102]. The bottom line is BookStack provides built-in, user-facing export options that cover the likely needs: PDF for printable docs, HTML/Markdown for editable transfers. This makes it easy to generate a quick manual or share content outside the system without hassle. It's a strong point of BookStack, considering many open source wikis require plugins or don't have as many formats.

Nextcloud: Nextcloud's Collectives is somewhat limited in formal export options. The Collectives FAQ explicitly was asked "When will HTML/PDF export be available" and the answer was: *"There's already an option to print a collective, which is meant as an easy shortcut to export it to PDF... See the 'Print' option in the menu... we decided against adding our own PDF export code for now since browser print-to-PDF exists."* [103]. So effectively, to get a PDF of a page or an entire collective, you use the browser's print dialog. If you choose the *Collective* (which is like the whole space) and hit Print, it will try to print all pages in that collective in one go – that might produce a PDF of multiple pages (the user in the FAQ noted the print menu is at collective level, not on each subpage) [104]. For a single page, you can of course print that page to PDF. For HTML, one can simply copy-paste or view source of the page. Also, since each page is stored as a Markdown file in Nextcloud, you could go to the Files app and just download the `.md` file (that's effectively an "export" in Markdown). If needed, you could then convert that markdown to HTML or PDF using external tools (like pandoc). In fact, a community member set up a "Flow" (Nextcloud automation) to convert markdown to PDF automatically [105] [64], and by Nextcloud 31 they introduced integration with **Pandoc** to allow exporting md to pdf/docx through the web interface if you install that app [63]. So things are improving. But out-of-the-box, it's rudimentary: the expectation is you'll use *Print to PDF*. Nextcloud doesn't offer a way to export to Word or any other formats natively for notes. This is a slight disadvantage if your team needs to regularly generate PDFs of documentation – it's doable, just not one-click except via the browser. On the import side, there's no dedicated importer for other platforms because Collectives just uses files: to "import", you could drag a bunch of Markdown files into the Collectives folder on the server and they become pages (though you might then need to add them to the collective via the database or UI unless Collectives auto-discovers files – not sure it does). Most likely, migration to Collectives would involve manually creating pages and copy-pasting content. On the plus side, because it's all open formats (markdown and files), you can script things if needed. But for a non-technical admin, Nextcloud lacks the nice export/import wizards the others have. Considering exporting is often needed for publishing or backup, this is a limitation. That said, if you already back up Nextcloud's storage and database, you inherently back up the notes too (so maybe they assume backups at system level rather than content export).

6. Hosting Compatibility (VMware, Kubernetes)

(Note: While not a direct user feature, this was explicitly asked, so we cover each tool's on-prem hosting considerations.)

DokuWiki: Very lightweight and flexible in deployment. In a VMware environment, one could spin up a small Linux VM (even 1 vCPU, 512MB RAM would do) with Apache or Nginx+PHP and just drop DokuWiki's files in. No database needed simplifies things. Backup is just the filesystem (the pages and media are in `/data`). For **containers/Kubernetes**, DokuWiki has an official Docker image on Docker Hub [65], which can be used in K8s. You'd need a persistent volume for the data directory. There are also charts or stack templates contributed by community. Because it's file-based, scaling horizontally is a bit tricky (you'd need shared storage or use it read-only scale-out with one write node), but that's usually not needed for a small team. So, DokuWiki is extremely easy to self-host and well-suited to on-prem. It's one of the least demanding options here.

Confluence: Confluence Data Center (the only on-prem option in 2026, since Server edition support ended in 2024) is a heavier enterprise application. To run it on VMware, typically you run it on a dedicated VM (or multiple VMs for a cluster). It requires Java (the installer bundles it) and a database like PostgreSQL or SQL Server. For a small team, a single node DC (which is allowed) on say 2-4 CPU, 8GB RAM VM would suffice. In Kubernetes, Atlassian provides Helm charts to deploy Confluence DC cluster, but that's generally for large installations – it's quite complex. You'd need to manage stateful sets, shared storage (for Confluence home directory or use their EFS recommendations), etc. It's doable but not trivial. Confluence is definitely **on-prem capable** – many enterprises run it internally. In fact, Atlassian specifically touts Data Center for those who cannot go cloud for data reasons [66]. The major issue is cost: Data Center licensing is annual and not cheap (especially if your user count grows). Also, Atlassian's strategic push is to cloud, but they committed support for Data Center at least till 2029 [106]. On the technical side, running Confluence requires a bit of admin skill (setting up the database, possibly proxy, etc.). But VMware is fine – you can even use the official Linux installer on a VM and you're set. For Kubernetes, it's possible but perhaps overkill for a small team – simpler to just run a VM. Summing up: Confluence can be self-hosted on your own infrastructure, albeit with higher resource requirements and complexity than the open-source options. Ensure you consider the licensing and maintenance overhead (patching, upgrades).

Docmost: Docmost is containerized by default (the documentation suggests Docker installation). It needs several components (app, DB, Redis). They likely provide a **Docker Compose** setup to run it easily. In VMware, you could either run that Compose on a Linux VM or break out the services onto different VMs (one for PostgreSQL, etc., depending on your preference). In Kubernetes, you would deploy it possibly as a helm chart or custom manifests (the company *Coolify* has some documentation on deploying Docmost, hinting that it's amenable to cloud-native deployment [107] [108]). Because it's a modern web app, containerization is natural for it. Running Docmost is more involved than PHP apps: you need to handle upgrades of the container, manage the PostgreSQL migrations, etc. But it's not too bad given Docker handles packaging. The **key** for on-prem is that Docmost is AGPL and you have full access to self-host; the only limitation is if you want certain features, you need a license key. On Kubernetes, you might have to create config maps or secrets for the environment variables for those enterprise features if you have them. Also, being relatively new, there may not be an official helm chart (though one might appear as adoption grows). So you might be writing your own deployment YAML or using Docker Compose on a VM. Either way, it's doable – just ensure you allocate enough resources. It's built with Node.js and React; performance should be fine on modest hardware, but real-time collab and indexing attachments (enterprise) means it can use CPU and memory as usage scales. A small team would probably be fine with 2 CPU/4GB for the app and a similarly small DB, to start. In summary, **on-prem hosting is fully supported** (the product is marketed for on-premise), with Docker making it relatively straightforward [68]. Just note you'll maintain a few services rather than a single LAMP stack.

BookStack: Very easy to self-host. For VMware: set up a LAMP (or LEMP) VM, or use the official **Ubuntu installation script** which automates installation of PHP, MySQL, etc. On Kubernetes: there is an official Docker image for BookStack, and popular third-party ones (LinuxServer.io has one). You'll also need a MySQL/MariaDB instance (which can be another container). The app doesn't need much RAM. Many run BookStack in Docker Compose with something like `linuxserver/bookstack` image + a MariaDB container + possibly a reverse proxy. In K8s, it's just wrapping those containers appropriately. People have contributed helm charts or you can adapt a generic chart. It's not complex to containerize since it's a single web service plus DB. Backups are straightforward (DB dump + storage volume for uploaded images). BookStack is also quite stable – upgrades are typically every few months and can be done via git pull or by replacing the Docker image. The **on-prem friendliness** is high: you don't need any external services. It doesn't phone home. It supports common environment (Apache or Nginx, MySQL, PHP) which fits well in typical enterprise Linux servers. So BookStack is well-suited to either a VM or container environment on-prem. Many small companies choose it exactly because it's simple to host internally. In Kubernetes, ensure persistence for uploads (though you can also use S3 for images if you prefer). But nothing unusual here – it's one of the easier apps to manage.

Nextcloud: Nextcloud can certainly be hosted on-prem and many organizations do so. On VMware, one might allocate a VM for Nextcloud – recommended specs depend on user count and usage, but e.g. 2-4 CPUs, 4-8GB RAM for a small team, plus storage for files. Nextcloud requires a web server (Apache or nginx with PHP), and a database (MariaDB/Postgres), plus typically Redis for caching file locks. The setup is more involved than BookStack because Nextcloud is a larger suite of features (think of it as hosting your own mini Google Suite). However, a lot of guides and even pre-made VM images (like the Nextcloud VM by T&M Hansson) exist to ease installation. In Kubernetes, there are community charts that deploy Nextcloud along with a database and other components. Nextcloud in K8s is doable but one must handle file storage (often via a Persistent Volume claim, possibly backed by NFS or similar, or use external S3 as primary storage). Some prefer to run Nextcloud on a VM for simplicity due to the heavy I/O nature. Running Nextcloud in Docker is also common (the official Docker image or the webdevops image, etc.). The **Collectives app** is just an app within Nextcloud; hosting Nextcloud covers it. So if you already have a Nextcloud server, enabling Collectives is trivial. If not, you'll be deploying a full Nextcloud instance just to serve as a documentation platform – which is perhaps overkill if you aren't using its other features. But if you also want the benefit of Nextcloud (file sharing, etc.), then it's a win-win. Nextcloud's on-prem readiness is one of its selling points: *"a modern, on-premises content collaboration platform"* [109]. It emphasizes data sovereignty – you keep everything in-house. The drawback is it's a heavier app to maintain (updates come every year majorly and need careful application; plus making sure you have backup of DB and storage is crucial). But plenty of support available via community and Nextcloud GmbH if needed. In Kubernetes, ensure to also run a OnlyOffice or Collabora container if you want Office integration (though not required for Collectives). Overall, Nextcloud can be comfortably hosted on-prem, fitting well in both VM and container paradigms, as long as you allocate appropriate resources and storage.

7. Strengths, Weaknesses, and Migration Considerations for OneNote Users

Finally, it's important to consider how each tool will feel for users migrating from **Microsoft OneNote**, especially regarding formatting habits and existing content migration:

- **OneNote vs Structured Wiki Paradigm:** OneNote is a *freeform* note-taking app – users can click anywhere on a page and create a text box, they can drag images and ink freely, and mix content without a rigid structure. None of the wiki/document tools can replicate that "floating canvas" approach [110]. They all enforce a linear flow of text (though Confluence allows multiple columns sections, it's still much more structured than OneNote). OneNote users might need to adjust to this constraint: instead of placing two text boxes side by side arbitrarily, they might use a table or a two-column layout macro (in Confluence) or create subsections. **This is a significant change in formatting habit.** OneNote also encourages quick jotting and inconsistent styling (since there's no enforced template) [111], whereas documentation systems often work best with more consistency (headings, etc.).

- **Formatting and Styling:** OneNote offers rich text formatting (colors, highlights, checkboxes, font sizes) very easily. Some wiki tools (Confluence, Docmost) allow colored text and highlight via macros or editor options; others (BookStack, DokuWiki, Nextcloud) either have limited or no support for arbitrary text colors/highlights out-of-the-box. Users who like to color-code or highlight in OneNote will find fewer options. Confluence has an `{info}` or text highlight feature, but e.g. BookStack's editor doesn't have a "text color" button by default. **Checkboxes/To-do lists:** OneNote has checkbox tags for making to-do lists within notes. Confluence supports checkbox lists (you can insert a task list and even assign people, and you can tick them off interactively) – this is actually an advantage of Confluence for those who used OneNote for task tracking [112]. BookStack and DokuWiki can have checkboxes as Unicode characters or as part of Markdown tasks (`- [ ]`), but those won't be interactive (except BookStack maybe just renders a box icon). Docmost might not have advanced task tracking yet (the feature request indicates it's not there currently) [113]. Nextcloud Text supports markdown checkboxes (`- [ ]` becomes a checkbox in preview and can be checked off) by Nextcloud 24 it should display them, but they might not persist state (they're just text toggled to [x]). So users coming from OneNote who used it for to-do lists might miss the integration; Confluence would come closest to providing an interactive checklist experience.

- **Embedded Files and Printouts:** In OneNote, a common practice is to *"print"* a document or PDF to OneNote, which places an image of each page of the PDF into the note for annotation or reading. None of the compared tools automatically do that kind of embedding. Instead, attachments are typically linked or previewed. Confluence can display an attached PDF within the page using the PDF macro (like a scrollable window of the PDF) – that might be a partial substitute. BookStack or DokuWiki do not have built-in PDF inline viewers, so a PDF would just be a link (user clicks and it downloads or opens externally). Nextcloud would integrate with a PDF viewer when clicked, but not show all pages inline on the wiki page. So if users liked having the content of attachments visible in the note itself, they will have to adjust: they might need to just keep attachments as separate or copy relevant content into the wiki page. Over time, the team might find this separation cleaner (OneNote printouts can make pages very long and heavy), but it's a change.

- **Hierarchy and Organization:** OneNote organizes notes as Notebooks > Sections > Pages (and optionally Section Groups and subpages). Each wiki here has its own hierarchy:

    - DokuWiki uses Namespaces > Pages (1–2 levels, effectively can simulate notebooks by namespaces).
    - Confluence uses Spaces > Pages > Sub-pages (any depth). This is somewhat analogous to Notebooks (Spaces) and a tree of pages inside. The concept of Section in OneNote is basically a subfolder in Confluence if you mimic it.
    - Docmost uses Workspaces (or Spaces) > Pages > sub-pages possibly. So similar to Confluence.
    - BookStack uses Shelves > Books > Chapters > Pages. This is a bit more structured than OneNote. Users coming from OneNote might find Shelves/Books either very helpful (if OneNote's Section groups were not enough) or a bit confining until they get used to it. It enforces at most 3 levels (Shelf is a grouping of books, but chapters cannot nest beyond one level of pages).
    - Nextcloud Collectives uses Collectives (think of these as notebooks) > Pages > subpages. That maps reasonably well to OneNote's Notebooks and pages. It might lack one intermediate level (OneNote sections), so large notebooks in OneNote might need to become multiple Collectives or use top-level pages as section-dividers.

  Migration-wise, users will need to decide how to map their OneNote structure to the new tool's structure. This can be a bit of manual planning but is usually straightforward for Confluence/Docmost (just treat OneNote Notebook as a Space, Sections as maybe top-level pages or as some categorization). For BookStack, perhaps each OneNote Notebook becomes a Book in BookStack, Section groups could be Shelves, sections as Chapters.

- **Learning Curve & Training:** OneNote is very informal – people just start typing anywhere. A wiki requires a little more discipline: you must choose a title for your page, place content in a linear fashion, use headings for structure, etc. Some users might initially find this restrictive, but in a documentation-heavy environment, this actually enforces better organization. It may be worth running a short training or providing templates so that OneNote users know how to do common tasks in the new tool (e.g., "How do I create a checklist? How to paste a screenshot? How to attach a file?"). In Confluence or BookStack, these tasks are pretty easy, but showing them once will help avoid frustration.

- **Migration of Existing Content:** OneNote does not export data very cleanly for import into these systems. Typically, you can export OneNote notebooks or sections to formats like PDF, DOCX, or MHT (web page) using the OneNote desktop app. There's no direct import in any of the compared tools for OneNote. So migration will likely be manual or semi-manual:

    - **Manual approach:** Open OneNote pages, copy content, paste into the wiki. This works if content is mostly text and images. You'll likely need to re-upload images when copying (some tools will copy images via clipboard too, e.g., Confluence might let you paste and it will create an attachment automatically).
    - **Semi-automated:** Export OneNote pages to Word documents, then use an importer (Confluence can import Word – preserving headings, bold, etc., fairly well [114] [115]). For BookStack or Docmost, one could export to docx, then convert docx to Markdown (tools like pandoc can do that) and import markdown. It's still a multi-step process. BookStack's

    community has an open request for OneNote import [102] but as of now it's not implemented. If you have hundreds of pages, some scripting may be worthwhile. For a smaller amount, copy-paste might be okay – though tedious, it ensures you reformat nicely.
    - Attachments would need to be saved from OneNote (OneNote attachments can be opened and saved out) then reattached to the new wiki pages.
    - **OneNote's freeform content issue:** If some OneNote pages have text boxes all over, when you copy them, the order might be jumbled or you might have to manually decide the flow. That requires a bit of editorial effort. It might be wise to simplify or clean up those notes during migration (which could be seen as an opportunity to improve the documentation structure).

- **User Acceptance:** Change can be challenging. OneNote has a very different feel – more like a personal notebook – whereas these tools feel more like formal documentation. It's good to communicate the benefits to the team:

    - *Single source of truth & search:* A wiki provides one place to search all notes, which can be more effective than OneNote's search (OneNote search can be limited, e.g., per section group, and not always reliable [116]). Tools like Confluence or Docmost have robust search engines, often full-text and typo-tolerant [116] [117] . This is a big win for finding info quickly.
    - *Access control & collaboration:* The new tools can ensure everyone is looking at the latest version (OneNote syncing can sometimes conflict if not on cloud, or if on SharePoint it can be clunky). With a wiki, there's one live page that everyone edits or reads. Permissions can also be more granular if needed (Confluence and BookStack can restrict who can edit/view certain spaces/books, etc., whereas OneNote was often all open or per notebook).
    - *Structured output:* Over time, having well-structured documentation (with consistent headings, etc.) can make it easier to produce PDFs or manuals, something not possible with OneNote directly (OneNote cannot export an entire notebook to a single PDF easily; you have to do section by section, which is a pain [118]). In the new system, exports are easier, enabling sharing of documentation outside the team.
    - *Integration:* Some of these tools integrate with other systems (Confluence with Jira, BookStack via webhooks or API, Docmost maybe with other platforms via embeds). Nextcloud ties into file storage. These integrations could smooth workflows, e.g., linking documentation to issue trackers or embedding content in chats.

- **Loss of Handwritten Notes:** If the team uses OneNote's ink features (handwriting or drawing with stylus), none of these wiki tools support that directly. They'd have to type, or attach scans of drawings. That's a loss of a feature. However, Docmost and BookStack integrate with draw.io/diagrams.net which can be a substitute for at least digital drawing of diagrams (though not the same as quick pen sketches). If people took handwritten notes on a tablet, they might need to switch to typing or attach those notes as images/PDFs. This might be worth noting in migration planning (the culture may shift to more typed, formal notes).

- **Offline Access:** OneNote allowed offline notebooks which sync later. Wiki tools are generally online-only (though Confluence has a read-only mode in mobile app offline for recently viewed, and BookStack/Docmost/Nextcloud have no official offline mode). So users will need network access to view/edit notes. In an on-prem setup, that usually means VPN if remote, etc. It's something to consider – if offline note-taking was a scenario (e.g., on a laptop offsite), they might have to jot notes in another form and input later.

- **Migration Volume & Strategy:** If there is a **large volume** of existing OneNote content, you may not want to dump it all blindly into the new system. This could be an opportunity to cull outdated info and restructure. It might be wise to manually migrate only important current documentation, and archive older notes as PDF somewhere for reference if needed. Each tool can accommodate large volumes, but user navigation experience is better if the content is well-organized rather than a direct dump.

In conclusion on migration concerns: **Users will need to adapt their note-taking style** to a more structured approach and expect a different editing experience. With training and time, most teams appreciate the benefits (especially the powerful collaboration and search features). Technically, migrating content requires some effort – there's no turnkey importer from OneNote, so plan for either manual copy-paste or intermediate conversion via Word/PDF. It's advisable to do a trial migration of a representative notebook to one of the candidate tools to gauge the effort and results, then proceed with the chosen platform accordingly.

Comparison Summary

To wrap up, here is a high-level summary of each tool's key strengths and weaknesses in this context:

- **DokuWiki:** *Strengths:* Very lightweight, easy on-prem deployment, no database needed, numerous plugins, and free LDAP/ACL support [119]. It's battle-tested and **reliable** for documentation [120]. *Weaknesses:* UI is dated and not as intuitive for non-technical users (markup editing by default) [121]. No real-time editing or modern collaboration features [1]. Formatting flexibility is limited compared to WYSIWYG editors. Migration from OneNote would be manual and formatting might need cleanup. Good for teams that value simplicity and control over flashy interface.

- **Confluence (Data Center):** *Strengths:* Rich feature set – excellent editor, powerful macros, best-in-class collaborative editing [2], attachments handling, and strong enterprise integration (LDAP, etc.). It's very **user-friendly** and familiar in feel to Office tools, which helps OneNote users transition. Also has the largest ecosystem (plugins for anything from diagrams to workflows). *Weaknesses:* **Cost** – requires a paid license that can be expensive for on-prem. Also resource-intensive to host. Some complexity in administration (upgrades, DB maintenance). Another soft consideration: Atlassian's push to cloud means long-term on-prem support is guaranteed only through 2029 [106], but that's still a while. Migration from OneNote still not automatic, but Confluence's Word import can ease part of it. For an organization willing to invest, Confluence provides a robust OneNote replacement with added benefits of structure and integration.

- **Docmost:** *Strengths:* Modern and feature-rich (almost a drop-in alternative to Confluence/Notion in open source form). It offers **real-time collaboration** [19], a slick UI, built-in diagramming, and Markdown support. It's also specifically designed for knowledge bases, with Spaces, page history, comments, etc. [4] [122]. On-prem deployment via Docker is relatively straightforward. *Weaknesses:* It's a newer project – still **early stage** [26], which might mean occasional bugs or missing minor features. The biggest consideration is that some enterprise features (especially **LDAP auth**) require a paid license [25]. If AD integration is a must and the budget is zero, that's a problem. Also, being new, its community and documentation depth is smaller than others. Migration from OneNote would be similar to Confluence (no direct import, but can leverage the Markdown/HTML import). If the team wants a Notion-like experience on-prem and can handle the enterprise feature cost (or doesn't mind local user accounts), Docmost is an attractive option.

- **BookStack:** *Strengths:* **Ease of use** – very intuitive for users of all skill levels [74]. The book/chapter/page structure is great for a documentation-heavy team to organize content logically. The WYSIWYG editor and Markdown option cover both bases, and built-in diagram integration is a plus [7]. It's open-source and free, including all features (LDAP, etc.). On-prem is simple (LAMP stack or Docker). It's relatively lightweight yet capable. Export options are excellent for a FOSS tool [56] [57]. *Weaknesses:* Lacks real-time concurrent editing (one editor at a time) [21], which could be a downside if the team frequently co-edits notes. Also, the rigid "Shelves & Books" structure might feel constrained if the team prefers a more ad-hoc organization (though one can also not use Shelves if not needed). It doesn't have an official mobile app, though that's not required. Overall, BookStack's **simplicity and user-friendliness** are its selling points, making it likely the least friction for OneNote users aside from the missing freeform canvas aspect.

- **Nextcloud (Collectives/Notes):** *Strengths:* Fully on-prem and **integrated platform** – if the team could benefit from other Nextcloud features (file sharing, etc.), this is a big plus. Collectives provides real-time co-editing in a Markdown environment [22], satisfying collaborative note-taking. It's open-source and includes AD integration for free [87]. Also, since notes are stored as files, it's easy to access or back them up, and even edit via other editors if needed. *Weaknesses:* The feature set for note-taking is **basic** – no advanced formatting beyond what Markdown offers [8]. Users might miss text highlighting, varied fonts, or more visual editing options. The UI, while clean, is not as polished or purpose-built for documentation as others (it's essentially a simple list of pages). There's also no built-in robust export, and navigation is limited to filtering and search, lacking cross-linking ease (though you can link pages manually). Another potential weakness is if you're not using Nextcloud for anything else, you'd be running a comparatively heavy system just for notes – a lot of admin overhead if you only need a wiki. Migration from OneNote would likely be manual as well, copying into markdown. Nextcloud suits an environment where you want a *lightweight wiki* tightly integrated with files and possibly where users already use Nextcloud.

Migration Concerns Recap: Regardless of tool, expect to invest time in restructuring and copying content from OneNote. OneNote's freeform notes must be linearized, and some formatting (like handwritten sections or arbitrary positioning) won't carry over [110]. Encourage users to embrace the new structure (use headings, use multiple pages instead of one huge canvas, etc.). There might be an adjustment period where users try to do something "the OneNote way" and it doesn't work – e.g., dragging an image next to text and it doesn't stay side by side. Training and documentation on "how to do X in the new tool" will alleviate this. On the flip side, they will soon find many advantages: for instance, no more wondering who has the latest version of a note, better search (especially in Confluence or Docmost, where search is quite powerful and maybe even OCRs or indexes attachments in enterprise versions), and the ability for multiple people to contribute easily rather than a single user's OneNote notebook.

Comparison Table Reference: For quick reference, the earlier comparison table captures many of these points with citations, which can be used to delve deeper into each statement.

Each tool can replace OneNote's core function of storing and organizing documentation, but they differ in approach: - If real-time collaboration and a modern interface are top priority, **Confluence or Docmost** lead (with the trade-off of cost for Confluence, or missing free LDAP for Docmost). - If a free, open-source solution with ease of use is paramount and you can live without concurrent editing, **BookStack** is an excellent choice (many describe it as a “friendly” Confluence alternative [123] [124]). - If the team is already inclined towards open-source and doesn’t mind a bit of wiki syntax for ultimate simplicity, **DokuWiki** is solid and reliable (though perhaps less “shiny” for end users). - If integration with broader file sharing or an existing Nextcloud deployment is desired, **Nextcloud Collectives** provides note-taking within that ecosystem, albeit with simpler editing features.

In conclusion, migrating from OneNote to any of these on-premise tools will bring more structured knowledge management to the team. By weighing the criteria above – especially editing experience, collaboration needs, and infrastructure fit – the team can choose the solution that best aligns with their workflow. Each alternative has its niche: from DokuWiki’s no-frills reliability [120] to Confluence’s all-in-one enterprise experience [2] [125]. Proper training and a thoughtful migration plan will ensure a successful transition, allowing the team to document efficiently with full control over their data.

---

[1] [5] [21] [26] [65] [67] [68] [69] [73] [74] [77] [82] [120] [121] [122] [123] [124] Choosing a Wiki for Small Business: 6 Options Reviewed
https://typemill.net/knowledge-hub/best-wiki-software

[2] [6] [10] [11] [13] [23] [66] [119] [125] Top 10 knowledge management tools | BlueSpice
https://bluespice.com/the-top-10-knowledge-management-tools/

[3] [4] [12] [14] [19] [20] [24] [25] [36] [54] [55] Introduction | Docmost - Documentation
https://docmost.com/docs/

[7] [72] Drawings & Diagrams · BookStack
https://www.bookstackapp.com/docs/user/diagrams/

[8] [75] [83] Nextcloud introduces collaborative rich text editor - Nextcloud
https://nextcloud.com/blog/nextcloud-introduces-collaborative-rich-text-editor/

[9] [61] [62] [94] [103] [104] Collectives FAQ - i Support - Nextcloud community
https://help.nextcloud.com/t/collectives-faq/135836

[15] [56] [57] [58] [59] [60] [100] [101] Exporting & Importing Content · BookStack
https://www.bookstackapp.com/docs/user/export-import/

[16] [17] [18] [78] [79] [80] [81] [106] Collaborative editing | Confluence Data Center 10.2 | Atlassian Documentation
https://confluence.atlassian.com/doc/collaborative-editing-858771779.html

[22] Nextcloud Collectives
https://nextcloud.github.io/collectives/usage/

[27] [28] [85] [86] LDAP Authentication · BookStack
https://www.bookstackapp.com/docs/admin/ldap-auth/

[29] [30] [87] User authentication with LDAP — Nextcloud latest Administration Manual latest documentation
https://docs.nextcloud.com/server/31/admin_manual/configuration_user/user_auth_ldap.html

[31] where do i put a file? - DokuWiki User Forum
https://forum.dokuwiki.org/d/6912-where-do-i-put-a-file

32 33 34 35 71 89 Upload Files | Confluence Data Center 10.0 | Atlassian Documentation
https://confluence.atlassian.com/doc/upload-files-139513.html

37 Docmost: The Wiki & Document Manager You Need to See! - YouTube
https://www.youtube.com/watch?v=wcK7iUNBUyo

38 39 40 41 42 90 91 92 Attachments · BookStack
https://www.bookstackapp.com/docs/user/attachments/

43 Copy images in Collectives - i Support - Nextcloud community
https://help.nextcloud.com/t/copy-images-in-collectives/133466

44 Page Export - DokuWiki
https://www.dokuwiki.org/export

45 Dw2Pdf plugin - DokuWiki
https://www.dokuwiki.org/plugin:dw2pdf

46 Microsoft OneNote - DokuWiki User Forum
https://forum.dokuwiki.org/d/14069-microsoft-onenote

47 48 49 50 51 52 53 97 98 99 Export Content to Word, PDF, HTML and XML | Confluence Data Center 10.2 | Atlassian Documentation
https://confluence.atlassian.com/doc/export-content-to-word-pdf-html-and-xml-139475.html

63 Ability to export to: Plain text, PDF, ODT, HTML, DOCX, ... · Issue #102
https://github.com/nextcloud/text/issues/102

64 Save as pdf from md text or notes? - i Support
https://help.nextcloud.com/t/save-as-pdf-from-md-text-or-notes/104720

70 109 Nextcloud - Open source content collaboration platform
https://nextcloud.com/content-collaboration-platform/

76 How to create Display Tables - i Support - Nextcloud community
https://help.nextcloud.com/t/how-to-create-display-tables/140074

84 LDAP Auth plugin: Active Directory - DokuWiki
https://www.dokuwiki.org/plugin:authldap:ad

88 How to Add files fo user download?
https://forum.dokuwiki.org/d/14268-how-to-add-files-fo-user-download

93 Nextcloud Collectives image problem - i Support
https://help.nextcloud.com/t/nextcloud-collectives-image-problem/221244

95 Files in Collectives Pages - i Support - Nextcloud community
https://help.nextcloud.com/t/files-in-collectives-pages/175458

96 tips:pdfexport [DokuWiki]
https://www.dokuwiki.org/tips:pdfexport

102 Import Export Function Word Documents and OneNote #5128 - GitHub
https://github.com/BookStackApp/BookStack/issues/5128

105 How do I convert markdown files to pdf files - i Support
https://help.nextcloud.com/t/how-do-i-convert-markdown-files-to-pdf-files/158324

[107] Docmost: Enterprise-ready Wiki for Teams
https://docmost.com/

[108] Best Self-Hosted Wiki Software - Docmost
https://docmost.com/blog/selfhosted-wiki-software/

[110] The Only OneNote Guide You'll Ever Need – Productivity Hub
https://productivityhub.org/2019/12/12/the-only-onenote-guide-youll-ever-need/

[111] [114] [115] [116] [117] [118] Perfect Wiki vs OneNote: What's the Better Wiki Solution for Microsoft Teams?
https://perfectwikiforteams.com/blog/perfect-wiki-vs-one-note/

[112] [113] [FR] Todo lists · Issue #629 · docmost/docmost - GitHub
https://github.com/docmost/docmost/issues/629