Markdown - Web App

Markdown ↔ HTML + WYSIWYG → Markdown app on Debian 12 using Nginx, from a blank server to a working site.

? One-Line Download & Execute:

```
clear && apt update && apt upgrade -y && apt install -y curl && clear && curl -s https://docs.greenhome.stream/attachments/45 | bash
```

What you'll build

- A static site served by **Nginx** that hosts a single index.html containing the full converter with a Quill toolbar for WYSIWYG editing.
- No backend or runtime is required; everything converts in the browser and is safe to serve as static files.

Prerequisites

- A Debian 12 system with shell access and basic sudo privileges to install packages and manage services.
- An IP on the LAN or WAN that can be reached from a browser, or localhost if testing locally.

Step 1 — Update packages

- Refresh APT metadata to ensure the latest repository view.

```
apt update
```

Step 2 — Install Nginx

- Install Nginx from Debian's repositories, which is sufficient for serving a static app.

```
apt install -y nginx
```

- Enable and start the service so it survives reboot and is immediately available.

```
systemctl enable --now nginx
```

- Optionally confirm status to verify the service is active (running).

```
systemctl status nginx
```

Step 3 — Create the web root

- Create a directory to keep the app isolated and easy to manage.

```
mkdir -p /var/www/markdown-app
```

- Set ownership to the web user for simple maintenance (optional for purely static reads).

```
chown -R www-data:www-data /var/www/markdown-app
```

Step 4 — Create the app file

- Open a new file named **index.html** under the web root.

```
nano /var/www/markdown-app/index.html
```

- Paste the app code with the Quill toolbar and the DOWNLOAD/EXPORT buttons), then save and exit.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Markdown ↔ HTML + WYSIWYG → Markdown/HTML</title>
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <!-- Quill Snow theme -->
  <link href="https://cdn.jsdelivr.net/npm/quill@1.3.7/dist/quill.snow.css" rel="stylesheet">
  <style>
    :root {
      --bg: #0f172a; --panel: #111827; --muted: #94a3b8; --text: #e5e7eb;
      --accent: #22c55e; --border: #1f2937; --btn: #1f2937; --btn-hover: #374151; --warn:
#f59e0b;
    }
    * { box-sizing: border-box; }
    body { margin: 0; background: var(--bg); color: var(--text); font: 14px/1.5 ui-sans-serif,
system-ui, -apple-system, Segoe UI, Roboto, Ubuntu, Cantarell, Noto Sans, Arial; }
    header { padding: 16px 20px; border-bottom: 1px solid var(--border); display:flex; align-
items:center; justify-content:space-between; background:#0b1220; }
    header h1 { font-size: 16px; margin: 0; font-weight: 600; }
    .mode { display:flex; gap:12px; align-items:center; color:var(--muted); flex-wrap:wrap; }
```

```
    .mode label { display:inline-flex; gap:6px; align-items:center; cursor:pointer; }
    main { height: calc(100dvh - 60px); display:grid; grid-template-columns: 1fr 1fr; gap:0; }
    .pane { display:grid; grid-template-rows: auto 1fr; min-height:0; border-right:1px solid
var(--border); }
    .pane:last-child { border-right:none; }
    .titlebar { display:flex; align-items:center; justify-content:space-between; padding:10px
12px; border-bottom:1px solid var(--border); background:#0b1220; }
    .titlebar h2 { margin:0; font-size:12px; letter-spacing:.04em; color:var(--muted); text-
transform:uppercase; }
    .actions { display:inline-flex; gap:8px; }
    button,.btn { border:1px solid var(--border); background:var(--btn); color:var(--text);
padding:8px 12px; border-radius:8px; cursor:pointer; font-weight:600; font-size:13px; }
    button:hover,.btn:hover { background:var(--btn-hover); }
    button.primary { border-color:#14532d; background:#14532d; }
    button.primary:hover { background:#166534; }

    /* PANE BODY: make editor area fill height, keep hint at bottom */
    .wrap {
      min-height: 0;
      padding: 10px;
      display: grid;
      grid-template-rows: 1fr auto; /* editor takes remaining height, hint sizes to content */
      gap: 8px;
    }

    /* Any visible input must stretch */
    #mdInput, #htmlInput, #quillWrap { height: 100%; }

    textarea, .editor, .ql-container { width:100%; }
    textarea {
      resize:none; border:1px solid var(--border); background:var(--panel); color:var(--text);
      border-radius:8px; padding:12px;
      font-family: ui-monospace, SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono",
"Courier New", monospace;
      height:100%;
    }
    textarea[readonly]{ opacity:.95; }
    .hint { color:var(--muted); font-size:12px; margin:0; }
    .warn { color:var(--warn); font-weight:600; }
    .pill { display:inline-block; padding:3px 8px; border-radius:999px; background:#1f2937;
```

```
border:1px solid var(--border); color:var(--muted); font-size:12px; }

    /* Quill container: full-height flex column (toolbar + editor) */
    #quillWrap {
      border:1px solid var(--border);
      border-radius:8px;
      background:#0f172a;
      overflow:hidden;
      display:flex;            /* becomes active when JS sets style.display = 'flex' */
      flex-direction:column;
      min-height:0;
    }
    #quillToolbar { flex: 0 0 auto; }
    #quillEditor   { flex: 1 1 auto; min-height:0; }

    /* Quill internals stretch */
    .ql-container { flex:1 1 auto; height:100%; }
    .ql-editor { color:#e5e7eb; height:100%; min-height:0; overflow:auto; }

    @media (max-width:980px) {
      main { grid-template-columns:1fr; height:auto; min-height:100dvh; }
      .pane { border-right:none; border-bottom:1px solid var(--border); }
      .pane:last-child { border-bottom:none; }
    }
  </style>
</head>
<body>
  <header>
    <h1>Markdown ↔ HTML + WYSIWYG → Markdown/HTML</h1>
    <div class="mode" role="radiogroup" aria-label="Conversion mode">
      <label><input type="radio" name="mode" value="md2html" checked /> Markdown →
HTML</label>
      <label><input type="radio" name="mode" value="html2md" /> HTML → Markdown</label>
      <label><input type="radio" name="mode" value="wysiwyg2md" /> WYSIWYG → Markdown</label>
      <label><input type="radio" name="mode" value="wysiwyg2html" /> WYSIWYG → HTML</label>
      <span class="pill">Auto-convert</span>
    </div>
  </header>

  <main>
```

```
    <!-- Left: inputs -->
    <section class="pane" id="leftPane">
      <div class="titlebar">
        <h2 id="leftTitle">Input</h2>
        <div class="actions">
          <button id="clearBtn" title="Clear input">Clear</button>
          <button id="sampleBtn" title="Insert sample">Sample</button>
        </div>
      </div>
      <div class="wrap" id="inputWrap">
        <!-- Markdown input -->
        <textarea id="mdInput" spellcheck="false" aria-label="Markdown input" placeholder="#
Hello
- Type Markdown on the left
- See HTML source on the right"></textarea>

        <!-- HTML input -->
        <textarea id="htmlInput" spellcheck="false" aria-label="HTML input" placeholder="<!--
Paste or type HTML here -->" style="display:none;"></textarea>

        <!-- WYSIWYG input (Quill) -->
        <div id="quillWrap" style="display:none;">
          <div id="quillToolbar">
            <span class="ql-formats">
              <select class="ql-header">
                <option selected></option>
                <option value="1"></option>
                <option value="2"></option>
                <option value="3"></option>
              </select>
              <select class="ql-font"></select>
              <select class="ql-size"></select>
            </span>
            <span class="ql-formats">
              <button class="ql-bold"></button>
              <button class="ql-italic"></button>
              <button class="ql-underline"></button>
              <button class="ql-strike"></button>
              <button class="ql-blockquote"></button>
              <button class="ql-code-block"></button>
```

```
            </span>
            <span class="ql-formats">
              <button class="ql-list" value="ordered"></button>
              <button class="ql-list" value="bullet"></button>
              <button class="ql-indent" value="-1"></button>
              <button class="ql-indent" value="+1"></button>
              <select class="ql-align"></select>
            </span>
            <span class="ql-formats">
              <button class="ql-link"></button>
              <button class="ql-image"></button>
            </span>
            <span class="ql-formats">
              <button class="ql-clean"></button>
            </span>
          </div>
          <div id="quillEditor"></div>
        </div>

        <p class="hint"><span class="warn">Note:</span> HTML produced from Markdown or the editor is sanitized before preview and export. </p>
      </div>
    </section>

    <!-- Right: outputs -->
    <section class="pane" id="rightPane">
      <div class="titlebar">
        <h2 id="rightTitle">Output</h2>
        <div class="actions">
          <button id="downloadBtn" title="Download Markdown (.md)">DOWNLOAD</button>
          <button id="exportBtn" class="primary" title="Export HTML (.html)">EXPORT</button>
        </div>
      </div>
      <div class="wrap">
        <textarea id="output" readonly spellcheck="false" aria-label="Conversion output"></textarea>
      </div>
    </section>
  </main>
```

```
  <!-- Libraries -->
  <script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/dompurify@3.0.9/dist/purify.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/turndown@7.2.0/dist/turndown.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/quill@1.3.7/dist/quill.min.js"></script>

  <script>
  (function() {
    // Elements
    const modeRadios = document.querySelectorAll('input[name="mode"]');
    const leftTitle = document.getElementById('leftTitle');
    const rightTitle = document.getElementById('rightTitle');

    const mdInput = document.getElementById('mdInput');
    const htmlInput = document.getElementById('htmlInput');
    const quillWrap = document.getElementById('quillWrap');
    const output = document.getElementById('output');

    const downloadBtn = document.getElementById('downloadBtn');
    const exportBtn = document.getElementById('exportBtn');
    const clearBtn = document.getElementById('clearBtn');
    const sampleBtn = document.getElementById('sampleBtn');

    // Converters
    const td = new TurndownService({ headingStyle: 'atx', codeBlockStyle: 'fenced' });

    // Configure Marked
    if (window.marked?.use) marked.use({ gfm: true, breaks: false, silent: true });

    // Quill initialization
    let quill = null;
    function ensureQuill() {
      if (quill) return quill;
      quill = new Quill('#quillEditor', { theme: 'snow', modules: { toolbar: '#quillToolbar' }
});
      quill.on('text-change', recalc);
      return quill;
    }

    // App state
```

```
    let mode = 'md2html';

    function setMode(next) {
      mode = next;
      // Toggle inputs (note: use 'flex' for Quill wrapper so it stretches)
      mdInput.style.display   = (mode === 'md2html') ? 'block' : 'none';
      htmlInput.style.display = (mode === 'html2md') ? 'block' : 'none';
      quillWrap.style.display = (mode === 'wysiwyg2md' || mode === 'wysiwyg2html') ? 'flex' :
'none';
      if (mode === 'wysiwyg2md' || mode === 'wysiwyg2html') ensureQuill();

      if (mode === 'md2html') {
        leftTitle.textContent = 'Input: Markdown';
        rightTitle.textContent = 'Output: HTML (source)';
      } else if (mode === 'html2md') {
        leftTitle.textContent = 'Input: HTML';
        rightTitle.textContent = 'Output: Markdown';
      } else if (mode === 'wysiwyg2md') {
        leftTitle.textContent = 'Input: WYSIWYG (toolbar)';
        rightTitle.textContent = 'Output: Markdown';
      } else {
        leftTitle.textContent = 'Input: WYSIWYG (toolbar)';
        rightTitle.textContent = 'Output: HTML (source)';
      }
      recalc();
    }

    function sanitize(html) {
      return window.DOMPurify ? DOMPurify.sanitize(html, { USE_PROFILES: { html: true } }) :
html;
    }

    function mdToHtmlSource(md) {
      const raw = (window.marked?.parse) ? marked.parse(md ?? '') : '';
      return sanitize(raw);
    }

    function htmlToMd(html) {
      return td.turndown(html ?? '');
    }
```

```
    function getQuillHtml() {
      if (!quill) return '';
      return quill.root.innerHTML || '';
    }

    function recalc() {
      if (mode === 'md2html') {
        output.value = mdToHtmlSource(mdInput.value);
      } else if (mode === 'html2md') {
        output.value = htmlToMd(htmlInput.value);
      } else if (mode === 'wysiwyg2md') {
        output.value = htmlToMd(getQuillHtml());
      } else if (mode === 'wysiwyg2html') {
        output.value = sanitize(getQuillHtml());
      }
    }

    function downloadFile(filename, content, mime) {
      const blob = new Blob([content], { type: mime || 'text/plain;charset=utf-8' });
      const url = URL.createObjectURL(blob);
      const a = document.createElement('a');
      a.href = url; a.download = filename; document.body.appendChild(a); a.click();
a.remove();
      URL.revokeObjectURL(url);
    }

    // Buttons
    downloadBtn.addEventListener('click', () => {
      // Always download Markdown (.md)
      if (mode === 'md2html') {
        downloadFile('document.md', mdInput.value || '', 'text/markdown;charset=utf-8');
      } else if (mode === 'html2md') {
        downloadFile('document.md', output.value || '', 'text/markdown;charset=utf-8');
      } else if (mode === 'wysiwyg2md') {
        downloadFile('document.md', output.value || '', 'text/markdown;charset=utf-8');
      } else if (mode === 'wysiwyg2html') {
        const md = htmlToMd(getQuillHtml());
        downloadFile('document.md', md, 'text/markdown;charset=utf-8');
      }
```

```
    });

    exportBtn.addEventListener('click', () => {
      // Always export HTML (.html)
      if (mode === 'md2html') {
        const htmlBody = mdToHtmlSource(mdInput.value || '');
        const full = `<!doctype html><html lang="en"><head><meta charset="utf-
8"><title>Export</title><meta name="viewport" content="width=device-width,initial-
scale=1"><style>body{font:16px/1.6 system-ui,-apple-system,Segoe
UI,Roboto,Arial;padding:24px;max-width:860px;margin:auto;}pre,code{font-family:ui-
monospace,SFMono-Regular,Menlo,Monaco,Consolas,Liberation Mono,Courier
New,monospace;}</style></head><body>${htmlBody}</body></html>`;
        downloadFile('document.html', full, 'text/html;charset=utf-8');
      } else if (mode === 'html2md') {
        const safe = sanitize(htmlInput.value || '');
        const full = `<!doctype html><meta charset="utf-8">${safe}`;
        downloadFile('document.html', full, 'text/html;charset=utf-8');
      } else if (mode === 'wysiwyg2md') {
        const safe = sanitize(getQuillHtml());
        const full = `<!doctype html><meta charset="utf-8">${safe}`;
        downloadFile('document.html', full, 'text/html;charset=utf-8');
      } else if (mode === 'wysiwyg2html') {
        const safe = sanitize(getQuillHtml());
        const full = `<!doctype html><meta charset="utf-8">${safe}`;
        downloadFile('document.html', full, 'text/html;charset=utf-8');
      }
    });

    clearBtn.addEventListener('click', () => {
      if (mode === 'md2html') mdInput.value = '';
      if (mode === 'html2md') htmlInput.value = '';
      if (mode === 'wysiwyg2md' || mode === 'wysiwyg2html') { if (quill) quill.setText(''); }
      output.value = '';
    });

    sampleBtn.addEventListener('click', () => {
      if (mode === 'md2html') {
        mdInput.value = `# Demo

- **Bold** and _italic_
```

```
- Code: \`console.log("hi")\`
- Link: [Example](https://example.com)

\`\`\`js
function hello(){ return "world"; }
\`\`\`
`;
      } else if (mode === 'html2md') {
        htmlInput.value = `<h1>Sample</h1><p><strong>Bold</strong> and
<em>italic</em></p><ul><li>A</li><li>B</li></ul>`;
      } else {
        ensureQuill();
        quill.root.innerHTML = `<h1>Sample</h1><p><strong>Bold</strong> and
<em>italic</em></p><ul><li>A</li><li>B</li></ul>`;
      }
      recalc();
    });

    // Mode change + live updates
    modeRadios.forEach(r => r.addEventListener('change', e => { if (e.target.checked)
setMode(e.target.value); }));
    mdInput.addEventListener('input', recalc);
    htmlInput.addEventListener('input', recalc);

    // Init
    setMode('md2html');
    mdInput.value = `# Welcome

Type Markdown on the left; view HTML source on the right.`;
    recalc();
  })();
  </script>
</body>
</html>
```

- The page loads libraries via CDN and runs entirely client-side, so no additional build or server components are required.

Step 5 — Configure an Nginx server block

- Create a dedicated server block that points root to the app directory and uses try_files for clean static handling.

```
nano /etc/nginx/sites-available/markdown-app
```

- Paste the config below.

```
server {
    listen 80;
    server_name _;

    root /var/www/markdown-app;
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }

    access_log /var/log/nginx/markdown-app.access.log;
    error_log  /var/log/nginx/markdown-app.error.log;
}
```

- Enable the site, remove the default, test, and reload Nginx.

```
ln -s /etc/nginx/sites-available/markdown-app /etc/nginx/sites-enabled/markdown-app
rm -f /etc/nginx/sites-enabled/default
nginx -t
systemctl reload nginx
```

Step 6 — (Optional) Open the firewall

- Debian may not ship UFW enabled by default, but if UFW is in use, allow HTTP or the "Nginx Full" profile.

```
# If UFW is not present:
apt install -y ufw

# Allow web traffic:
ufw allow 'Nginx HTTP'     # or: sudo ufw allow 80/tcp
ufw status
```

- If UFW was disabled and needs enabling, do so carefully on remote systems to avoid lockouts.

```
ufw enable
```

Step 7 — Test the app

- From a browser on the same network or the host, open http://SERVER_IP and the converter UI should appear immediately.
- Switch among “Markdown → HTML”, “HTML → Markdown”, and “WYSIWYG → Markdown”, and use the right-side DOWNLOAD (.md) and EXPORT (.html) buttons.

Maintenance notes

- To update the app, edit /var/www/markdown-app/index.html and reload the page; no service restart is required for static assets.
- Access and error logs for this site are written to the paths defined in the server block and help with troubleshooting.
- If a newer mainline Nginx is ever required, installing from the official nginx.org packages is supported with repository configuration.

Troubleshooting

- Validate configuration if Nginx refuses to reload: nginx -t outputs syntax or path errors for quick fixes.
- Check service health with systemctl status nginx and look for “active (running)” to confirm it started correctly.
- Use the site’s access and error logs or journalctl -u nginx to diagnose 404s or permission issues.

Downloads

- index.html.bak # Rename the file by removing the last '.bak'

Optional: One?shot replacement of the default site

- As a quick test without creating a new server block, replacing the default document root also works for simple local setups.

```
cp /var/www/html/index.nginx-debian.html /var/www/html/index.nginx-debian.html.bak
cp /var/www/markdown-app/index.html /var/www/html/index.html
systemctl reload nginx
```

- Visiting the server IP should then render the app from the default site if the root was switched as shown.

You’re done

- The app now runs as a static site on Debian 12 with **Nginx**, ready to be bookmarked or documented in BookStack.

Ins0mniA

Revision #13
Created 2025-09-02 02:11:22 EEST by Green
Updated 2025-09-04 02:12:22 EEST by Green