Author Quick Start — Simple-CMS Prototype

One-page guide for authors to create, validate, collaborate, and publish lessons.

1) Open repository in VS Code

- File → Open Folder → select repository root.

- Accept recommended extensions (YAML, Markdown, markdownlint, Live Share).

2) Scaffold a lesson (recommended)

```
# scaffold and open in VS Code
python .\scripts\create_lesson.py "My Lesson Title" --course Microsoft-Secu
```

Or copy the template manually:

```
mkdir -Force Content\Microsoft-Security\Module-01
Copy-Item Templates\lesson-template.md Content\Microsoft-Security\Module-01
code Content\Microsoft-Security\Module-01\my-lesson-001.md
```

3) Edit frontmatter and content

- Update YAML frontmatter fields: `id`, `title`, `course`, `module`, `status` (Draft | Review | Approved | Published), `authors`.

- Save frequently. Use Live Share for pair editing.

4) Validate metadata and sync index

```
python .\scripts\check_status_consistency.py --fix
```

5) When ready for review

- Update frontmatter to `status: "Review"` and push a branch or invite reviewers via Live Share.

6) Approve and publish

When a reviewer decides the lesson is final, run:

```
python .\scripts\workflow_transition.py <article-id> Approved
```

This will move the file into `Content/Published/` and update `content_status.json` to `Published` automatically.

7) Commit & open PR

```
git add Content\<course>\<module>\*.md content_status.json
git commit -m "Add/update <article-id>"
git push --set-upstream origin demo/<your-branch>
# Open PR via GitHub web UI or use `gh pr create` (if you have GitHub CLI c
```

Notes

- Use branches for parallel work or Live Share for real-time collaboration.

- If the checker reports parse errors, install `pyyaml` in the environment used by `python`:

```
python -m pip install pyyaml
```

If you want, I can create and open the PR for you (I need `gh` auth or a GitHub token), or you can click the PR creation link that I've opened in your browser.

Author Quick Checklist

A short printable checklist authors can use when creating, reviewing, and publishing content.

Scaffold

Run: `python scripts/create_lesson.py "<Title>" --course "<Course>" --module "<Module>" --open`

Confirm file created under `Content/<Course>/<Module>/<slug>.md` with YAML frontmatter.

Drafting

Edit content in the `.md` file; keep frontmatter correct and complete.

6. Set `status: "Draft"` while drafting.

Run local linters and validator if present; install pre-commit hook: `.\scripts\install_precommit_hook.ps1`.

Pre-commit checks

Run: `python scripts/check_status_consistency.py --fix` to normalize statuses and update `content_status.json`.

Create PR

Branch and push changes; open Pull Request for review.

Ensure CI checks (frontmatter checks, markdownlint) pass.

Address Feedback

Make requested changes and push to the PR branch until reviewers approve.

Final Approval & Publish

After approval, run:

17. `python scripts/workflow_transition.py <slug> Approved --by "your.name@example.com"`

    This records `approved_by`/`approved_at`, updates `content_status.json`, and moves the file to `Content/Published/`.

    Post-publish

    Verify file in `Content/Published/` and that `content_status.json` shows `current_status: "Published"`.

21. Delete your feature branch if desired:

22. `git push origin --delete <branch>`

23. `git branch -D <branch>`

Notes

- Do not rely on folder placement for status — frontmatter is canonical.

- Keep binaries out of `main`; use GitHub Releases for distribution assets.

- For troubleshooting, see `WORKFLOW.md` or run the consistency checks.

Content Authoring & Publishing Workflow

This document finalizes the frontmatter-first workflow for the repository and documents the end-to-end steps from creating a topic to publishing it.

Principles - Frontmatter-first: YAML frontmatter in each markdown file is the single source of truth for metadata (`title`, `slug`, `course`, `module`, `status`, etc.). - Single edit tree: Authors edit files under `Content/<Course>/...`; only `Content/Published/` is used to hold published artifacts. - CI & index driven: `content_status.json` is derived from frontmatter and CI enforces correctness.

Status values - Draft, Review, Approved, Published - Only `Approved` triggers an automatic move to `Content/Published/` via `scripts/workflow_transition.py`.

Commands (common) - Scaffold a lesson (create file and frontmatter):

```
python scripts/create_lesson.py "Customer Onboarding" --course "Microsoft-D
```

- Rebuild/sync index from frontmatter:

```
python scripts/check_status_consistency.py --fix --from-frontmatter
```

- Run a status transition (preferred entrypoint):

```
# Record approver and publish when Approved
python scripts/workflow_transition.py customer-onboarding Approved --by "a
```

- Synchronize a single file's status to/from the index:

```
python scripts/sync_status.py Content/Microsoft-Dynamics/Module-02/customer
```

Review & PR process 1. Create a feature branch for your edits. 2. Run `check_status_consistency.py` locally before pushing. 3. Open a PR and request reviewers. 4. Address feedback and update the file (frontmatter remains canonical). 5. Once reviewers approve, run `workflow_transition.py <slug> Approved --by "approver"` to publish.

Audit fields - On approval the script records `approved_by` and `approved_at` in `content_status.json`; when the file is moved these fields are also appended to the markdown frontmatter.

Deprecated behavior - Folder-based staging (Draft/Review/Approval) and multiple stage mover scripts have been removed; the single canonical entrypoint is `scripts/workflow_transition.py`.

Troubleshooting - If a file does not move on approval, check script output and `content_status.json` for updated `path` and `current_status`. - Use `git log` / `git revert` to roll back accidental moves.

Contact / support - For authoring questions, ask the docs team or open an issue in this repo.