STARTER KIT: Build Your UDA System from Scratch

A step-by-step guide to instantiate Universal Documentation Architecture for any domain.

Follow this workflow to build a production-ready documentation system in 1-2 weeks.

PHASE 0: Pre-Work (1-2 hours)

Do this before you touch any files.

0.1: Read the Doctrine (MANDATORY)

Before you proceed, study the DOCUMENTATION_DOCTRINE. *(TOC below)*

This doctrine defines the non-negotiable rules that keep UDA systems maintainable.
Violating them breaks the system. The Starter Kit assumes you follow these rules.

> **Attention**
>
> *The non-negotiable rules that make UDA systems work.*

Rule	Condensed Statement
1. One Page = One Intent	Each page answers exactly one question.
2. Strict Routing Discipline	Route every page to one section using the matrix.
3. No Mixing Modes	A page is either Task, Concept, or Reference (never blended).
4. No Narrative or History	Keep pages timeless; history goes in ADRs or Release Notes.
5. No Miscellaneous Sections	Split content until it fits a template.
6. Templates Required	Use one of the 8 canonical templates, no exceptions.
7. Metadata Mandatory	Every page has author, last_verified, status, plus type-specific fields.
8. Cross-Linking Discipline	Follow routing rules; avoid circular or Task-to-Task links.
9. Event-Driven Updates	Update docs when systems change, not on a calendar.
10. Ownership in Metadata	Each section has an owner; each page has a steward.
11. Half-Life Reviews	Auto-flag pages past their freshness threshold.
12. Accessibility	Clear titles, no jargon without glossary, text + images, formatted code.
13. Verifiability	Every fact must match current system state.
14. No Duplication	One source of truth; link instead of repeating.
15. Standard Folder Structure	Use numeric prefixes for predictable ordering.
16. Folder READMEs	Each section explains scope, boundaries, and usage.
17. Central Templates	All templates live in `00-Start/Templates/`.

> **Danger**
>
> **Final Rule:** If you break the rules, you break the system.

0.2 Learn the routing matrix

Here's a **condensed routing table** version of the doctrine's routing discipline.

User Intent	Section	Page Type	Allowed Links →
How do I…?	Tasks	Task	Reference, Concepts, Troubleshooting
What is…?	Concepts	Concept	Concepts, Reference
Exact fields/flags?	Reference	Reference	Concepts
How do I fix…?	Troubleshooting	Troubleshooting	Tasks, Concepts, Reference
Operate/maintain?	Platform Ops	Runbook	Reference, Concepts, Troubleshooting
What changed?	Release & Upgrade	Release Notes	ADRs, Concepts
Why did we choose…?	ADRs	ADR	ADRs, Release Notes, Concepts
What does this mean?	Glossary	Glossary	Concepts

0.2 Decide Your Domain

Question: What domain will this KB cover?

Examples:

- Kubernetes cluster operations
- Python library documentation
- REST API reference
- Product user guide
- Internal procedures
- Framework documentation

Write it down. Everything flows from this decision.

0.3 Identify Your Users

Questions:

- Who is the primary audience? (Beginner / Intermediate / Advanced)
- What problems are they trying to solve?
- What 5-10 questions do they ask most?

Example (Kubernetes KB):

- Primary: Platform engineers and SREs

- Problems: Deploying, troubleshooting, upgrading, securing
- Top questions:
    1. How do I deploy an application?
    2. Why is my Pod not starting?
    3. What's a Service?
    4. How do I upgrade the cluster?
    5. How do I set up RBAC?

Write these down. These become your seed pages.

0.4 Choose Your Top-Level Sections

Rule: Pick 7-11 sections (Miller's Law - cognitive load limit).

#	Standard (11)	API (7)	Software (8)
1	Start Here	Start Here	Start Here
2	Tutorials	Getting Started	Onboarding
3	Tasks	How-To Guides	How-To Guides
4	Concepts	Concepts	Concepts
5	Reference	API Reference	Reference
6	Troubleshooting	Troubleshooting	Troubleshooting
7	Platform Operations	Glossary	FAQs
8	Security & Compliance	—	Release Notes
9	Releases & Upgrades	—	—
10	Architecture Decisions	—	—
11	Glossary	—	—

This way you have a single table with **row numbers** for quick reference, while each column represents one complete pattern.

Would you like me to also prepare a **blank version** of this table (with just numbers and empty cells) so you can immediately fill in your own custom section set alongside these three?

Your decision: List your sections and write a one-line description for each.

PHASE 1: Structure (2-3 hours)

Now build the folder structure and templates.

1.1 Create Folder Structure

Create these folders (use numeric prefixes for automatic ordering):

```
your-kb/
├── 00-Start/
├── 10-Tutorials/
├── 20-Tasks/
├── 30-Concepts/
├── 40-Reference/
├── 50-Troubleshooting/
├── 60-PlatformOps/
├── 70-Security/
├── 80-ReleaseUpgrade/
├── 90-ADRs/
└── 99-Glossary/
```

Note: If you chose fewer sections, skip ranges but keep the numeric structure. Example for a 7-section API KB:

```
├── 00-Start/
├── 10-GettingStarted/
├── 20-HowTos/
├── 30-Concepts/
├── 40-Reference/
├── 50-Troubleshooting/
└── 99-Glossary/
```

1.2 Create Templates/ Folder

Create a template for each section:

- Template - Task.md
- Template - Concept.md
- Template - Reference.md
- Template - Troubleshooting.md
- Template - Runbook.md
- Template - Release Notes.md
- Template - ADR.md
- Template - Tutorial.md

1.3 Customize Templates for Your Domain

Copy these 8 templates from UDA - Template Section 4. Customize section headings but keep the structure.

Example customizations:

For **Kubernetes**:

- "Task" template keeps all sections as-is
- "Concept" template adds `audience` metadata (Beginner/Intermediate/Advanced)
- "Reference" template adds `version_min` and `version_max` fields

For **REST API**:

- "Reference" template adds endpoint URL, HTTP method, authentication requirements
- "Task" template replaces "Common Pitfalls" with "Common Errors"

For **Python library**:

- "Task" template adds "Import statements" section
- "Reference" template adds "Type hints" section

Rule: Customize section headings and add domain-specific metadata, but keep the core structure intact.

1.4 Create Operational Documents

Create these documents in 00-Start/:

Welcome & Conventions.md

- Overview of the KB structure
- How to navigate it
- Key terminology
- Links to this Starter Kit and the Doctrine

Maintenance Workflow - Creating & Updating Pages.md

- When to create a new page
- Which template to use (create a routing matrix table)
- How to verify content
- How to mark stale pages
- Link to DOCUMENTATION_DOCTRINE

Creating a New Page - Checklist.md

- Step-by-step checklist for creating pages
- Link to template files
- Validation steps
- Example of a complete page

Pages Pending Review.md

- (Optional) Dataview dashboard showing pages with `status: needs_review` or exceeding half-life
- Only if you use Dataview plugin

PHASE 2: Create Main README (30 min)

Create README.md at the root of your KB:

```
# [Your KB Name]

_A [domain] knowledge base built on Universal Documentation Architecture._

## Getting Started

- **New here?** Start with [[00-Start/Welcome & Conventions]]
- **Looking for something?** Use Obsidian search (Ctrl+Shift+F)
- **Want to contribute?** See [[00-Start/Creating a New Page - Checklist]]

## Structure

This KB is organized by user intent, not topic. Choose based on what you're trying
to do:

- [[10-Tutorials]] - Guided walkthroughs for learning
- [[20-Tasks]] - How-to guides (quick answers)
- [[30-Concepts]] - Explanatory material (understand the why)
- [[40-Reference]] - Exact syntax and parameters
- [[50-Troubleshooting]] - Diagnosis and fixes
- [[60-PlatformOps]] - Operational runbooks
- [[70-Security]] - Security and compliance
- [[80-ReleaseUpgrade]] - Release notes and upgrades
- [[90-ADRs]] - Why we made certain decisions
- [[99-Glossary]] - Terminology and definitions

## Governance

- **Questions?** See [[DOCUMENTATION_DOCTRINE]] for the rules
- **Confused?** See [[00-Start/Maintenance Workflow - Creating & Updating Pages]]
- **Want to contribute?** Follow [[00-Start/Creating a New Page - Checklist]]

---

Built with [UDA](link to UDA - Template.md if applicable).
```

PHASE 3: Add README to Each Section (1 hour)

Create README.md in each top-level folder. Here's the template pattern (customize section names for your domain):

Template README Pattern

```
# [Section Name]

[One-sentence description of what this section covers]

## What This Section Is For

[Explain the user intent that routes to this section]

## When to Use This Section

- [Specific use case 1]
- [Specific use case 2]
- [Specific use case 3]

## When to Use Other Sections Instead

- [Other intent] -> [[XX-OtherSection]]
- [Other intent] -> [[XX-OtherSection]]

## How Pages Are Structured

[Explain the template used in this section and why]

## Example Topics

[List 3-5 example pages that belong in this section]
```

Example: Reference README

```
# Reference

Complete documentation of APIs, commands, and parameters.

## What This Section Is For

Lookup tool for questions like "What are the exact fields/flags/options?" You
already know what you need; you just need correct syntax.

## When to Use This Section

- Need exact syntax or parameter names
```

```
- Looking up API field specifications
- Verifying allowed values or constraints
- Want all options in one place

## When to Use Other Sections Instead

- Want to learn how it works -> [[30-Concepts]]
- Need step-by-step instructions -> [[20-Tasks]]
- Something isn't working -> [[50-Troubleshooting]]

## How Pages Are Structured

Each reference page is a terse lookup tool (not a tutorial). It includes:
- Syntax/signature
- All parameters with types and defaults
- Examples of real usage
- Links to related concepts

## Example Topics

- API endpoint parameters
- Command-line flags
- Configuration file schema
- Field reference for data structures
```

PHASE 4: Seed Content (4-8 hours)

Create your first 5-10 pages using the templates.

4.1 Pick Your Seed Topics

Go back to your list of "5-10 most common user questions" from Phase 0.

Example (Kubernetes KB):

1. How do I deploy an application? (Task)
2. Why is my Pod not starting? (Troubleshooting)
3. What's a Service? (Concept)
4. How do I upgrade the cluster? (Runbook)
5. How do I set up RBAC? (Task)
6. Pod API reference (Reference)
7. Service API reference (Reference)
8. How does networking work? (Concept)

4.2 Create One Seed Page Per Template

Create at least one page for each template type to test them:

- 1 Task page
- 1 Concept page
- 1 Reference page
- 1 Troubleshooting page
- 1 Runbook page
- 1 Release Notes page (optional for v1)
- 1 ADR page (optional for v1)
- 1 Glossary entry

4.3 Follow the Template Exactly

Copy the template, fill in the content, verify all sections are present.

4.4 Add Metadata

Every page must have frontmatter:

```
---
author: [Your Name or Team]
last_verified: YYYY-MM-DD
status: current
---
```

Add domain-specific metadata:

- Concepts: `audience: Beginner | Intermediate | Advanced`
- Troubleshooting:
  - `symptom_keywords: [comma, separated, list]`
  - `severity: common | rare`
- Reference: `version_min: X.Y` and `version_max: X.Y`
- Tasks: `prerequisites: [list of required knowledge]`

4.5 Cross-Link Using Routing Rules

Link pages according to DOCUMENTATION_DOCTRINE > 2. Strict Routing Discipline (Diataxis):

Task pages link to:

- Reference (for exact syntax)
- Concepts (for understanding)
- Troubleshooting (for common errors)
- Never other Tasks

Concept pages link to:

- Concepts (related concepts only)
- Reference (for specs)
- Never Tasks

Troubleshooting pages link to:

- Tasks (for fixes)
- Concepts (for background)
- Reference (for exact fields)

Use Obsidian wikilinks: `[[Page Name]]`

4.6 Verify All Links Resolve

In Obsidian, use "Find unlinked references" to catch broken links.

PHASE 5: Validation (1-2 hours)

Before launch, verify the system works.

5.1 Template Validation Checklist

For each seed page, verify:

- [ ] Uses correct template (Task, Concept, Reference, etc.)
- [ ] All template sections present
- [ ] Frontmatter metadata complete
- [ ] Title is clear and searchable
- [ ] Content is terse (no unnecessary explanation)
- [ ] Examples are concrete and runnable
- [ ] Links follow routing discipline
- [ ] No broken links (Obsidian unlinked references)

5.2 Routing Validation

For each seed page, ask:

- [ ] Is this in the right section?
- [ ] Does the title match user intent?
- [ ] Could I find this page if I were the target user?

5.3 Metadata Validation

For each seed page, verify:

- [ ] `author` field is present
- [ ] `last_verified` is today's date
- [ ] `status` is one of: current, needs_review, stale, superseded
- [ ] Domain-specific metadata is complete

5.4 Navigation Validation

Starting from 00-Start/Welcome & Conventions:

- [ ] Can you navigate to every folder?
- [ ] Does each folder README explain its purpose?
- [ ] Can you find seed pages by browsing?
- [ ] Can you find seed pages by searching?

PHASE 6: Launch (30 min)

6.1 Assign Section Owners

For each top-level folder, assign one owner:
copy to 00-Start/Maintenance Workflow

Section	Owner	Review Cadence
00-Start	.	Monthly
10-Tutorials	.	Quarterly
20-Tasks	.	Monthly
30-Concepts	.	Quarterly
40-Reference	.	Event-driven (releases)
50-Troubleshooting	.	Event-driven (incidents)
60-PlatformOps	.	Monthly
70-Security	.	Event-driven (CVEs)
80-ReleaseUpgrade	.	Event-driven (releases)
90-ADRs	.	As decisions are made
99-Glossary	.	Quarterly

6.2 Brief Owners on Maintenance

Owners should know:

- How to verify content (test pages, check links)
- When to flag pages as stale
- How to handle contributions
- Half-life expectations for their section

6.3 Announce to Users

- Link in team docs / Slack / internal wiki
- Brief message: "This KB is now live. Start with 00-Start/Welcome & Conventions"
- Encourage feedback

PHASE 7: Grow Continuously (Ongoing)

7.1 Add Pages Incrementally

Each time you answer a user question, consider creating a page:

- Follow the routing matrix
- Use the appropriate template
- Add frontmatter metadata
- Cross-link using routing rules
- Assign to section owner for verification

7.2 Maintenance Loop

Here's how you can turn your **Maintenance Loop** into a practical **checklist + Dataview dashboards** inside Obsidian. The idea is:

- Each task becomes a checklist item.
- Each cadence (Weekly, Monthly, Quarterly, Event-driven) gets its own Dataview query to surface the relevant pages automatically. See: DASHBOARDS

Weekly

- [ ] Review pages flagged `needs_review`
- [ ] Update pages affected by incidents

Monthly

- [ ] Section owner verifies critical pages
- [ ] Check for broken links
- [ ] Update stale metadata

Quarterly

- [ ] Full section review
- [ ] Verify routing is still correct
- [ ] Archive superseded pages

Event-driven

- [ ] New release: Update Reference, Release Notes, Task pages
- [ ] CVE: Flag Troubleshooting, Runbooks, Security pages
- [ ] Incident: Update Troubleshooting same day
- [ ] Major change: Create ADR, update related pages

How to Use

- Place this checklist in your **Maintenance Workflow** doc.
- Each cadence has its own Dataview block that auto-populates based on metadata (`status` , folder location, etc.).
- Owners tick off the checklist items after reviewing the dashboard outputs.

Would you like me to also add **frontmatter examples** (e.g., `status: needs_review` , `last_verified: YYYY-MM-DD`) so your dashboards will populate correctly right away?

7.3 Manage Rot

Every page has a `last_verified` date and expected half-life. When `last_verified` exceeds the half-life:

1. Flag page with `status: needs_review`
2. Assign to section owner
3. Owner verifies accuracy or marks stale
4. Update `last_verified` date

TROUBLESHOOTING

"I don't know what template to use"

Use the routing matrix in DOCUMENTATION_DOCTRINE Section 2. Answer these questions:

1. What is the user trying to do/understand/fix?
2. Does this match one of the 8 intents in the matrix?
3. Use the template in that row.

If it doesn't match one of the 8 intents, it needs to be split into multiple pages.

"This content doesn't fit any template"

This is a signal that:

- You're mixing modes (should be split into separate pages), OR
- It's a catch-all that violates the doctrine

Split it into separate pages that each fit one template.

"Users can't find my pages"

- Are the folder READMEs clear about scope?
- Are page titles searchable (do they start with the user's intent)?
- Are links correct in the routing matrix?
- Check 00-Start/Welcome & Conventions - is it clear how to navigate?

"I have too many pages in one section"

This is good! Means the section is active. Split large sections:

- Instead of 20-Tasks, create 20-GettingStarted and 21-AdvancedTasks
- Adjust numbering accordingly
- Keep numeric prefixes consistent

NEXT STEPS

1. ☐ **Go to Phase 0** and answer the questions.
2. ☐ **Go to Phase 1** and build the structure.
3. ☐ **Go to Phase 2** and create the main README.
4. ☐ **Go to Phase 3** and add section READMEs.
5. ☐ **Go to Phase 4** and create seed content.
6. ☐ **Go to Phase 5** and validate.

7. ☐ **Go to Phase 6** and launch.
8. ☐ **Go to Phase 7** and grow continuously.
