Confluence Knowledge Base Integration Guide

Complete guide for setting up, populating, and maintaining Confluence as the central knowledge repository for AI-powered support.

> ⚠️ **Important**: This integration uses **HTTP Request nodes** with the Confluence REST API, not dedicated Confluence nodes. All examples use Basic Auth with Atlassian email + API token.

Table of Contents

Overview

Confluence serves as the central knowledge repository for the AI support system. The system uses a hybrid approach:

1. **Confluence** - Stores human-readable documentation, runbooks, and solutions
2. **Vector Embeddings** - Enables semantic search across all Confluence content
3. **AI Synthesis** - Claude reads relevant pages and generates custom solutions

How It Works

```
┌─────────────────────────────────────────────────────────┐
│ 1. INITIAL INDEXING (One-time setup)                    │
│                                                          │
│ Confluence Pages → OpenAI Embeddings → Supabase Vector DB│
│                                                          │
└─────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────┐
│ 2. SUPPORT REQUEST RESOLUTION (Real-time)               │
│                                                          │
│ Customer Issue → Generate Query Embedding → Search Vectors│
│       ↓                                                  │
│ Top 5 Confluence Page IDs → Fetch Full Content → AI Reads  │
│       ↓                                                  │
│ AI Synthesizes Solution → Send to Customer                 │
│                                                          │
└────────────────────────────────────────────────────────────┘

┌────────────────────────────────────────────────────────────┐
│ 3. CONTINUOUS LEARNING (Daily)                             │
│                                                          │
│ Analyze Resolutions → Identify Gaps → Create New Pages     │
│       ↓                                                  │
│ Update Existing Pages → Re-index → Improve Future Searches │
│                                                          │
└────────────────────────────────────────────────────────────┘
```

Prerequisites

Before setting up Confluence integration:

- ⍰ Confluence Cloud or Data Center account
- ⍰ Admin access to create spaces and manage permissions
- ⍰ API token or OAuth credentials for n8n
- ⍰ Supabase account with pgvector enabled
- ⍰ OpenAI API key for embeddings
- ⍰ n8n instance with Confluence integration installed

Initial Setup

Step 1: Create Confluence API Credentials

For Confluence Cloud:

1. Go to https://id.atlassian.com/manage-profile/security/api-tokens
2. Click **Create API token**
3. Name it: `n8n Support System`
4. Copy the token (you won't see it again)

For Confluence Data Center:

1. Go to Settings ⟶ Personal Access Tokens
2. Create new token with scopes:
   - `READ` - confluence
   - `WRITE` - confluence
3. Copy the token

Step 2: Configure n8n Credentials

1. Open n8n UI
2. Go to **Settings** → **Credentials**
3. Click **Add Credential** → **HTTP Basic Auth**
4. Enter:
   - **Name**: `Confluence API (Basic Auth)`
   - **User**: Your Atlassian account email
   - **Password**: Paste the API token from Step 1
5. Click **Save**

> **Note**: The Confluence integration uses HTTP Request nodes with the REST API. You'll configure the base URL (`https://your-company.atlassian.net/wiki/rest/api/content`) directly in each workflow node.

Step 3: Set Up Supabase Vector Store

1. Sign up at supabase.com
2. Create a new project
3. Go to **SQL Editor** and run:

```sql
-- Enable vector extension
CREATE EXTENSION IF NOT EXISTS vector;

-- Create knowledge base table
CREATE TABLE confluence_kb (
  id BIGSERIAL PRIMARY KEY,
  page_id TEXT UNIQUE NOT NULL,
  space_key TEXT NOT NULL,
  title TEXT NOT NULL,
  content TEXT NOT NULL,
  url TEXT NOT NULL,
  embedding VECTOR(1536), -- OpenAI text-embedding-3-small dimension
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  last_indexed_at TIMESTAMPTZ DEFAULT NOW()
);

-- Create index for vector similarity search
CREATE INDEX ON confluence_kb USING ivfflat (embedding vector_cosine_ops)
WITH (lists = 100);

-- Create index for page_id lookups
CREATE INDEX idx_confluence_page_id ON confluence_kb(page_id);

-- Create index for space filtering
CREATE INDEX idx_confluence_space_key ON confluence_kb(space_key);
```

```sql
-- Function to search similar documents
CREATE OR REPLACE FUNCTION match_confluence_pages(
  query_embedding VECTOR(1536),
  match_threshold FLOAT DEFAULT 0.7,
  match_count INT DEFAULT 5
)
RETURNS TABLE (
  page_id TEXT,
  title TEXT,
  content TEXT,
  url TEXT,
  similarity FLOAT
)
LANGUAGE SQL STABLE
AS $$
  SELECT
    page_id,
    title,
    content,
    url,
    1 - (embedding <=> query_embedding) AS similarity
  FROM confluence_kb
  WHERE 1 - (embedding <=> query_embedding) > match_threshold
  ORDER BY embedding <=> query_embedding
  LIMIT match_count;
$$;
```

4. Go to **Settings** → **API** and copy:
   - Project URL
   - `anon` public key
   - `service_role` secret key
5. Configure in n8n:
   - Add **Supabase** credential
   - Enter URL and service role key

Confluence Space Structure

Recommended Organization

Create a dedicated Confluence space for support knowledge:

Space Key: `PKB` (Projects Knowledge Base) **Space Name**: `Projects Knowledge Base`

> **Note**: The CX-Catalyst system uses the `PKB` space for AI-generated KB articles. Ensure this space exists before running Workflow 5.

```
Support Knowledge Base
|
├── 📁 Getting Started
│   └── About This Knowledge Base
│
├── 📁 Authentication & Access
│   ├── Password Reset Guide
│   ├── SSO Configuration
│   ├── MFA Setup Instructions
│   └── API Key Management
│
├── 📁 Billing & Subscriptions
│   ├── Payment Method Update
│   ├── Invoice Access
│   ├── Subscription Changes
│   └── Refund Process
│
├── 📁 Product Configuration
│   ├── Initial Setup Guide
│   ├── Integration Configuration
│   ├── Webhook Setup
│   └── Advanced Settings
│
├── 📁 Troubleshooting
│   ├── Common Error Codes
│   ├── Performance Issues
│   ├── Connection Problems
│   └── Data Sync Issues
│
├── 📁 Known Issues & Bugs
│   ├── Current Incidents
│   ├── Planned Maintenance
│   └── Resolved Issues Archive
│
└── 📁 Internal Runbooks
    ├── Escalation Procedures
    ├── Emergency Response
    └── System Architecture
```

Page Templates

Create Confluence templates for consistency:

Template 1: Solution Article

```
## Problem Statement
[Clear description of the issue customers face]

## Affected Users
- Product: [Which product/plan]
- Environment: [Production/Staging/All]
- Frequency: [Common/Occasional/Rare]
```

```
## Solution

### Quick Fix
[Immediate workaround if available]

### Step-by-Step Resolution
1. [First step with clear instructions]
2. [Second step]
3. [Verification step]

### Expected Outcome
[What customer should see after completing steps]

## Prerequisites
- [Required access levels]
- [Required tools or permissions]
- [Any dependencies]

## Related Articles
- [Link to related Confluence page]
- [Link to product documentation]

## Metadata
- **Category**: [Authentication/Billing/Configuration/etc.]
- **Priority**: [Critical/High/Medium/Low]
- **Last Updated**: [Date]
- **Verified By**: [Name]

---
*AI-Searchable: Yes*
*Public: [Yes/No/Internal Only]*
```

Template 2: Runbook

```
## Overview
[Brief description of when to use this runbook]

## Scope
- **Severity**: [Critical/High/Medium/Low]
- **Response Time**: [Immediate/1 hour/4 hours/24 hours]
- **Escalation Path**: [Team/Individual to notify]

## Pre-Checks
- [ ] Verify issue exists
- [ ] Check system status page
- [ ] Review recent changes

## Investigation Steps
1. [Check logs]
```

```
2. [Review metrics]
3. [Verify configuration]

## Resolution Procedure
1. [Step with rollback plan]
2. [Step with validation]
3. [Final verification]

## Post-Resolution
- [ ] Notify affected customers
- [ ] Update status page
- [ ] Document in incident log
- [ ] Create postmortem (if critical)

## Rollback Plan
[How to revert changes if resolution fails]

## Contacts
- **Primary**: [Name/Slack handle]
- **Secondary**: [Name/Slack handle]
- **Management**: [Name/Slack handle]
```

Creating Knowledge Base Content

Initial Content Population

1. Audit Existing Knowledge

Gather from: - ✅ Previous support tickets with solutions - ✅ Internal documentation - ✅ Product documentation - ✅ Team wiki pages - ✅ Email threads with common solutions - ✅ Slack conversations with resolutions

2. Prioritize by Frequency

Create pages for: 1. **Top 20 most common issues** (80% of volume) 2. **Critical/emergency procedures** 3. **Product-specific configurations** 4. **Known bugs and workarounds** 5. **New feature documentation**

3. Content Guidelines

Write for AI and Humans: - ✅ Clear, descriptive titles - ✅ Structured with headers - ✅ Step-by-step instructions - ✅ Include error messages verbatim - ✅ Add screenshots and diagrams - ✅ Use consistent terminology - ❌ Avoid vague language - ❌ Don't use "click here" without context - ❌ Avoid assuming prior knowledge

SEO/Search Optimization: - Include common search terms - Add alternative phrasings - List related error codes - Include product/feature names

4. Labeling Strategy
Use Confluence labels consistently:

Product Labels: - `product:web-portal` - `product:mobile-app` - `product:api` - `product:enterprise`

Category Labels: - `category:authentication` - `category:billing` - `category:configuration` - `category:bug` - `category:feature`

Priority Labels: - `priority:critical` - `priority:high` - `priority:medium` - `priority:low`

Status Labels: - `status:current` - `status:archived` - `status:under-review`

Indexing Confluence Pages

Create Indexing Workflow in n8n

Create a new workflow: **Confluence KB Indexer**

Workflow Structure:

```
[Manual Trigger / Schedule]
         ↓
[Get All Pages from Space]
    (Confluence Node)
         ↓
[Loop Through Pages]
         ↓
[Get Page Content]
    (Confluence Node)
         ↓
[Clean & Format Content]
    (Code Node)
         ↓
[Generate Embedding]
    (OpenAI Node)
         ↓
[Upsert to Supabase]
    (Supabase Node)
         ↓
[Log Progress]
```

Node Configuration:

1. Confluence: Get Pages

```json
{
  "space": "PKB",
  "limit": 100,
  "expand": "body.storage,version,metadata.labels"
}
```

2. Code: Clean Content

```
// Remove HTML tags and format for embedding
const content = $input.item.json.body.storage.value;
const title = $input.item.json.title;
const pageId = $input.item.json.id;
const spaceKey = $input.item.json.space.key;
const url = `https://your-company.atlassian.net/wiki/spaces/${spaceKey}/pages/
${pageId}`;

// Strip HTML
const cleanContent = content
  .replace(/<[^>]*>/g, ' ')
  .replace(/\s+/g, ' ')
  .trim();

// Create searchable text: title + content
const searchableText = `${title}\n\n${cleanContent}`;

// Truncate if too long (max 8000 tokens ≈ 32000 chars)
const truncated = searchableText.substring(0, 32000);

return {
  page_id: pageId,
  space_key: spaceKey,
  title: title,
  content: truncated,
  url: url,
  metadata: {
    labels: $input.item.json.metadata.labels,
    version: $input.item.json.version.number,
    lastModified: $input.item.json.version.when
  }
};
```

3. OpenAI: Generate Embedding

```
{
  "model": "text-embedding-3-small",
  "input": "={{ $json.content }}"
}
```

4. Supabase: Upsert

```
INSERT INTO confluence_kb (
  page_id,
  space_key,
  title,
  content,
  url,
  embedding,
  metadata,
  last_indexed_at
)
VALUES (
  '{{ $json.page_id }}',
  '{{ $json.space_key }}',
  '{{ $json.title }}',
  '{{ $json.content }}',
  '{{ $json.url }}',
  '{{ $json.embedding }}',
  '{{ $json.metadata }}'::jsonb,
  NOW()
)
ON CONFLICT (page_id)
DO UPDATE SET
  title = EXCLUDED.title,
  content = EXCLUDED.content,
  url = EXCLUDED.url,
  embedding = EXCLUDED.embedding,
  metadata = EXCLUDED.metadata,
  updated_at = NOW(),
  last_indexed_at = NOW();
```

Run Initial Index

1. Activate the Confluence KB Indexer workflow
2. Click **Execute Workflow**
3. Monitor execution (may take 5-10 minutes for 100+ pages)
4. Verify in Supabase:

   ```
   SELECT COUNT(*) FROM confluence_kb;
   SELECT page_id, title FROM confluence_kb LIMIT 10;
   ```

Schedule Regular Re-indexing

Set the workflow to run: - **Daily at 2 AM** - Full re-index to catch updates - **Hourly (optional)** - For frequently updated pages

---

Workflow Integration

Modify Workflow 2: Self-Service Resolution

Update the workflow to query Confluence:

Add Vector Search Node

Position: After "Classify Issue" and before "Generate Solution"

Node: HTTP Request to Supabase

```
{
  "method": "POST",
  "url": "{{ $env.SUPABASE_URL }}/rest/v1/rpc/match_confluence_pages",
```

```
  "authentication": "predefinedCredentialType",
  "nodeCredentialType": "supabaseApi",
  "headers": {
    "apikey": "={{ $credentials.supabaseApi.serviceRole }}",
    "Content-Type": "application/json"
  },
  "body": {
    "query_embedding": "={{ $json.issue_embedding }}",
    "match_threshold": 0.7,
    "match_count": 5
  }
}
```

Update AI Agent Node

Add retrieved Confluence content to the prompt:

```
You are a customer support AI assistant.

Customer Issue:
{{ $json.description }}

Classification:
- Category: {{ $json.category }}
- Priority: {{ $json.priority }}
- Product: {{ $json.product }}

Relevant Knowledge Base Articles:
{{ $json.confluence_results.map(r => `
Title: ${r.title}
URL: ${r.url}
Similarity: ${r.similarity.toFixed(2)}
Content: ${r.content.substring(0, 1000)}...
`).join('\n---\n') }}

Based on the customer's issue and the relevant KB articles above, provide:
1. A clear, personalized solution
2. Step-by-step instructions
3. Links to the most relevant KB articles
4. Any prerequisites or warnings

Keep the response concise and actionable.
```

Modify Workflow 5: Continuous Learning

Add Confluence update capabilities:

Check for Knowledge Gaps

```
// Analyze cases without KB matches
const casesWithoutKB = $items.filter(item =>
  item.json.kb_match_score < 0.7 &&
  item.json.resolution_successful === true
);

// Group by category
const gaps = casesWithoutKB.reduce((acc, item) => {
  const category = item.json.category;
  if (!acc[category]) acc[category] = [];
  acc[category].push({
    description: item.json.description,
    solution: item.json.resolution,
    frequency: 1
  });
  return acc;
}, {});

return Object.entries(gaps).map(([category, issues]) => ({
  category,
  issue_count: issues.length,
  sample_issues: issues.slice(0, 5)
}));
```

Create New Confluence Pages

Node: AI Agent - Draft KB Article

```
Based on these resolved support cases that have no KB article:

Category: {{ $json.category }}
Number of cases: {{ $json.issue_count }}

Sample issues:
{{ $json.sample_issues.map(i => `- ${i.description}\n  Solution: ${i.solution}
`).join('\n') }}

Create a KB article using this template:

## Problem Statement
[Describe the issue]

## Solution
[Step-by-step instructions]

## Related Information
[Any additional context]

Format the output as Confluence storage format (HTML).
```

Node: HTTP Request - Create Confluence Page

Important: Use HTTP Request node with Basic Auth, not the dedicated Confluence node. Pre-escape content in a Code node first.

Code Node: Prepare Confluence Content

```
// Pre-escape content for JSON embedding
const jsonSafeContent = content
  .replace(/\\/g, '\\\\')
  .replace(/"/g, '\\"')
  .replace(/\n/g, '\\n')
  .replace(/\r/g, '\\r')
  .replace(/\t/g, '\\t');

return {
  json: {
    article_content_escaped: jsonSafeContent,
    article_title_escaped: title.replace(/"/g, '\\"')
  }
};
```

HTTP Request Node Configuration: - Method: POST - URL: `https://your-company.atlassian.net/wiki/rest/api/content` - Authentication: HTTP Basic Auth (email + API token) - Body (JSON - without = prefix):

```
{
  "type": "page",
  "title": "{{ $json.article_title_escaped }}",
  "space": {
    "key": "PKB"
  },
  "body": {
    "storage": {
      "value": "{{ $json.article_content_escaped }}",
      "representation": "storage"
    }
  },
  "status": "current"
}
```

Add Labels Node (HTTP Request): - Method: POST - URL: `https://your-company.atlassian.net/wiki/rest/api/content/{{ $json.id }}/label` - Body:

```
[
  {"prefix": "global", "name": "ai-generated"},
  {"prefix": "global", "name": "kb-article"},
  {"prefix": "global", "name": "needs-review"}
]
```

Re-index New Pages

After creating pages, trigger the Confluence KB Indexer workflow:

Node: HTTP Request

```
{
  "method": "POST",
  "url": "{{ $env.N8N_WEBHOOK_BASE_URL }}/webhook/confluence/reindex",
  "body": {
    "page_ids": "={{ $json.created_pages.map(p => p.id) }}"
  }
}
```

Automatic Updates

Update Existing Pages

When Workflow 5 identifies improvements:

1. **Fetch existing page**
2. **AI suggests edits**
3. **Update page content**
4. **Re-index**

Node: Confluence - Get Page

```
{
  "pageId": "{{ $json.confluence_page_id }}",
  "expand": "body.storage,version"
}
```

Node: AI Agent - Suggest Updates

```
Current KB article:
Title: {{ $json.title }}
Content: {{ $json.body.storage.value }}

Recent support cases suggest this article is missing:
{{ $json.missing_info }}

Suggest updates to the article. Provide the complete updated content in
Confluence storage format.
```

Node: Confluence - Update Page

```
{
  "pageId": "{{ $json.page_id }}",
  "version": {
    "number": "={{ $json.current_version + 1 }}"
  },
  "title": "={{ $json.title }}",
  "body": {
    "storage": {
      "value": "={{ $json.updated_content }}",
      "representation": "storage"
    }
  }
}
```

Maintenance

Weekly Tasks

Monday: Review KB Performance

```sql
-- Pages with most search hits
SELECT
  page_id,
  title,
  COUNT(*) as search_hits
FROM kb_search_log
WHERE searched_at > NOW() - INTERVAL '7 days'
GROUP BY page_id, title
ORDER BY search_hits DESC
LIMIT 20;

-- Pages never retrieved
SELECT page_id, title, url
FROM confluence_kb
WHERE page_id NOT IN (
  SELECT DISTINCT page_id
  FROM kb_search_log
)
AND created_at < NOW() - INTERVAL '30 days';
```

Wednesday: Content Freshness Check

```sql
-- Pages not updated in 90 days
SELECT
  page_id,
  title,
  url,
  updated_at,
  AGE(NOW(), updated_at) as age
FROM confluence_kb
WHERE updated_at < NOW() - INTERVAL '90 days'
ORDER BY updated_at ASC;
```

Friday: Gap Analysis Run Workflow 5 manually to identify: - Missing KB articles - Low-performing articles (searched but low satisfaction) - Duplicate or conflicting articles

Monthly Tasks

Review and Archive - Archive outdated pages (add `status:archived` label) - Remove deprecated product features - Consolidate similar articles - Update screenshots and examples

Quality Audit - Check top 50 pages for accuracy - Verify all links work - Ensure consistent formatting - Review AI-generated content

Performance Optimization

```sql
-- Rebuild vector index if needed
REINDEX INDEX confluence_kb_embedding_idx;

-- Update statistics
ANALYZE confluence_kb;

-- Check index usage
SELECT
  schemaname,
  tablename,
  indexname,
  idx_scan,
  idx_tup_read,
  idx_tup_fetch
FROM pg_stat_user_indexes
WHERE tablename = 'confluence_kb';
```

Best Practices

Content Writing

1. **Be Specific with Titles**
   - ✅ "How to Reset Password for SSO Users in Production"
   - ❌ "Password Reset"
2. **Include Variations**
   - Add a "Also Known As" section with alternative terms
   - Include common misspellings
   - List related error codes
3. **Use Real Examples**
   - Show actual error messages
   - Include sample API requests/responses
   - Provide concrete values, not placeholders
4. **Link Generously**
   - Link to related Confluence pages
   - Link to product documentation
   - Link to external resources
5. **Maintain Metadata**
   - Keep labels current
   - Update "Last Reviewed" dates
   - Mark deprecated content

Search Optimization

1. **Front-Load Important Terms**
   - Put key terms in the title and first paragraph
   - Use descriptive headers
2. **Avoid Jargon in Titles**
   - Titles should be customer-facing language
   - Explain acronyms in content
3. **Include Context**
   - Mention product names
   - Specify which versions are affected
   - Note environment (production, staging)

AI Interaction

1. **Structure for Parsing**
   - Use consistent header levels
   - Put solutions in clear sections
   - Use numbered lists for steps
2. **Provide Complete Information**
   - Include prerequisites
   - List assumptions
   - Mention edge cases
3. **Update Based on Feedback**
   - Monitor which pages AI retrieves
   - Check resolution success rates
   - Refine content based on patterns

Troubleshooting

Issue: Vector Search Returns No Results

Symptoms: - AI generates generic solutions - No Confluence pages in context - `kb_match_score: 0` in logs

Diagnosis:

```sql
-- Check if pages are indexed
SELECT COUNT(*) FROM confluence_kb;

-- Check if embeddings exist
SELECT COUNT(*) FROM confluence_kb WHERE embedding IS NOT NULL;

-- Test manual search
SELECT * FROM match_confluence_pages(
  (SELECT embedding FROM confluence_kb LIMIT 1),
  0.5,
  5
);
```

Solutions: 1. Re-run Confluence KB Indexer workflow 2. Check OpenAI API quota 3. Verify Supabase connection 4. Lower `match_threshold` to 0.6 or 0.5

Issue: Outdated Content Retrieved

Symptoms: - Solutions reference old features - Deprecated instructions provided - Customers report incorrect information

Solutions: 1. Add “Last Verified” date to pages 2. Filter by `updated_at` in search: `sql    WHERE updated_at > NOW() - INTERVAL '180 days'` 3. Archive old pages instead of deleting 4. Use version labels (`v1-deprecated`, `v2-current`)

Issue: Duplicate Articles

Symptoms: - Multiple pages for same issue - Conflicting instructions - AI retrieves wrong page

Solutions: 1. Search for duplicates: `sql    SELECT title, COUNT(*)    FROM confluence_kb    GROUP BY title    HAVING COUNT(*) > 1;` 2. Merge pages and redirect 3. Use canonical links 4. Update to reference single source of truth

Issue: Poor AI Synthesis

Symptoms: - AI ignores retrieved KB content - Solutions don’t match context - Generic responses despite good matches

Solutions: 1. Increase context in prompt 2. Show full page content (not truncated) 3. Explicitly instruct AI to prioritize KB 4. Add examples of good synthesis

Issue: Slow Indexing

Symptoms: - Indexer workflow times out - Takes hours to index 100 pages - High OpenAI API costs

Solutions: 1. Batch API calls (10-20 at a time) 2. Use `text-embedding-3-small` (cheaper, faster) 3. Cache embeddings (only re-index if content changed) 4. Index only modified pages: `sql    WHERE updated_at > last_indexed_at`

Metrics to Track

Monitor these in your dashboard:

Coverage Metrics

- Total Confluence pages indexed
- Pages created by AI this month
- Percentage of issues with KB match

Quality Metrics

- Average similarity score for retrieved pages
- Resolution success rate (with KB vs without)

- Customer satisfaction by KB article

Usage Metrics
- Top 20 most-retrieved pages
- Pages never retrieved (candidates for archive)
- Search queries with no results

Performance Metrics
- Average vector search latency
- Embedding generation time
- Index size and growth rate

KB Coverage by Customer Tier

The knowledge base currently contains **100+ articles** organized by customer tier:

Customer Tier	Article Count	Key Categories
Enterprise	50	API & Integration (15), Security & Compliance (10), SSO/Auth (8), Performance & Scaling (7), Admin & Config (10)
SMB	30	Account Management (8), Billing & Subscriptions (6), Integrations (8), Product Features (8)
Small Business	20	Order Management (5), Shipping & Returns (4), Account Help (3), Product FAQs (3), Getting Started (5)

Articles are stored in the Confluence `PKB` space and indexed into the `confluence_kb` Supabase table with vector embeddings for semantic search.

To expand coverage, see the Best Practices Guide for article structure, tagging, and content freshness guidelines.

Next Steps

1. **Complete Initial Setup** (Today)
   - ? Create Confluence space
   - ? Set up API credentials
   - ? Configure Supabase
2. **Populate Knowledge Base** (Week 1)

- [?] Create 50-100 core articles across customer tiers (see KB Coverage below)
- [?] Apply templates and labels
- [?] Review and edit for consistency

3. **Index and Test** (Week 1)
   - [?] Run initial indexing workflow
   - [?] Test vector search
   - [?] Verify AI can retrieve pages
4. **Integrate Workflows** (Week 2)
   - [?] Update Workflow 2 with KB search
   - [?] Test end-to-end resolution
   - [?] Monitor resolution quality
5. **Enable Automatic Updates** (Week 3)
   - [?] Configure Workflow 5 to create pages
   - [?] Set up scheduled re-indexing
   - [?] Implement feedback loop
6. **Ongoing Optimization** (Monthly)
   - [?] Review KB performance
   - [?] Archive outdated content
   - [?] Expand coverage based on gaps

Questions? - Workflow issues: Check n8n execution logs - Confluence problems: Check API connection - Vector search issues: Review Supabase logs - AI synthesis problems: Review agent prompts

---

Confluence Integration Guide v2.0 - January 2026