

N8n + AI Agent Content Reactor System

Architecture Overview



Core Workflow Structure

1. N8n Master Workflow: Content Processing Pipeline

json

```
{
  "name": "Content Reactor Master Pipeline",
  "nodes": [
    {
      "name": "Content Upload Trigger",
      "type": "@n8n/nodes-base.webhook",
      "parameters": {
        "path": "content-upload",
        "responseMode": "responseNode"
      }
    },
    {
      "name": "AI Agent: Analyze Content",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "http://localhost:8000/ai-agent/analyze-content",
        "method": "POST",
        "body": {
          "content_url": "{{${$json.content_url}}",
          "metadata": "{{${$json.metadata}}",
          "task": "full_analysis"
        }
      }
    },
    {
      "name": "Upload to NotebookLM",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "https://notebooklm.googleapis.com/v1/notebooks/{{${$env.NOTEBOOK_ID}}/documents",
        "authentication": "oAuth2Api"
      }
    },
    {
      "name": "AI Agent: Generate Content Strategy",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "http://localhost:8000/ai-agent/generate-strategy",
        "body": {
          "notebook_context": "{{${$json.notebook_response}}",
          "content_analysis": "{{${$json.analysis}}",
          "platforms": ["tiktok", "instagram", "linkedin", "twitter", "youtube"]
        }
      }
    }
  ]
}
```

```

    }
  }
},
{
  "name": "Split by Platform",
  "type": "@n8n/nodes-base.splitInBatches",
  "parameters": {
    "batchSize": 1
  }
},
{
  "name": "AI Agent: Create Platform Content",
  "type": "@n8n/nodes-base.httpRequest",
  "parameters": {
    "url": "http://localhost:8000/ai-agent/create-content",
    "body": {
      "platform": "{{json.platform}}",
      "content_suggestions": "{{json.suggestions}}",
      "brand_guidelines": "{{env.BRAND_GUIDELINES}}"
    }
  }
},
{
  "name": "Generate Media Assets",
  "type": "@n8n/nodes-base.switch",
  "parameters": {
    "rules": [
      {
        "condition": "{{json.platform === 'tiktok'}}",
        "output": 0
      },
      {
        "condition": "{{json.platform === 'instagram'}}",
        "output": 1
      }
    ]
  }
}
]
}

```

2. AI Agent API Endpoints

python

```

from fastapi import FastAPI, BackgroundTasks
from pydantic import BaseModel
import asyncio
from typing import List, Dict, Any

app = FastAPI()

class AIContentAgent:
    def __init__(self):
        self.notebooklm_client = NotebookLMClient()
        self.content_analyzer = SemanticAnalyzer()
        self.strategy_generator = ContentStrategyAI()
        self.media_generator = MediaCreationAI()
        self.performance_tracker = PerformanceTracker()

# API Models
class ContentAnalysisRequest(BaseModel):
    content_url: str
    metadata: Dict[str, Any]
    task: str

class StrategyRequest(BaseModel):
    notebook_context: str
    content_analysis: Dict[str, Any]
    platforms: List[str]

class ContentCreationRequest(BaseModel):
    platform: str
    content_suggestions: List[Dict]
    brand_guidelines: Dict[str, Any]

# Core AI Agent Endpoints
@app.post("/ai-agent/analyze-content")
async def analyze_content(request: ContentAnalysisRequest):
    """
    AI Agent analyzes incoming content and extracts semantic insights
    """
    agent = AIContentAgent()

# Download and transcribe content
transcript = await agent.transcribe_content(request.content_url)

```

```
# Semantic analysis
```

```
analysis = await agent.content_analyzer.full_analysis(  
    transcript=transcript,  
    metadata=request.metadata  
)
```

```
# Identify key moments and viral potential
```

```
moments = await agent.content_analyzer.identify_key_moments(transcript)
```

```
return {  
    "transcript": transcript,  
    "analysis": analysis,  
    "key_moments": moments,  
    "viral_potential_score": analysis.viral_score,  
    "recommended_platforms": analysis.platform_recommendations  
}
```

```
@app.post("/ai-agent/generate-strategy")
```

```
async def generate_content_strategy(request: StrategyRequest):
```

```
    """
```

```
    AI Agent creates platform-specific content strategy using NotebookLM context
```

```
    """
```

```
    agent = AIContentAgent()
```

```
# Query NotebookLM for contextual insights
```

```
context_insights = await agent.notebooklm_client.query_for_content_strategy(  
    request.notebook_context  
)
```

```
# Generate platform-specific strategies
```

```
strategies = {}
```

```
for platform in request.platforms:
```

```
    strategy = await agent.strategy_generator.create_platform_strategy(  
        platform=platform,  
        content_analysis=request.content_analysis,  
        context_insights=context_insights,  
        historical_performance=await agent.performance_tracker.get_platform_performance(platform)  
    )  
    strategies[platform] = strategy
```

```
return {
```

```

        "strategies": strategies,
        "priority_queue": agent.strategy_generator.prioritize_content(strategies),
        "optimal_posting_schedule": agent.strategy_generator.calculate_posting_schedule(strategies)
    }

@app.post("/ai-agent/create-content")
async def create_platform_content(request: ContentCreationRequest):
    """
    AI Agent creates actual content assets for specific platform
    """
    agent = AIContentAgent()

    content_assets = []
    for suggestion in request.content_suggestions:
        # Generate the content piece
        asset = await agent.media_generator.create_content_asset(
            platform=request.platform,
            suggestion=suggestion,
            brand_guidelines=request.brand_guidelines
        )

        content_assets.append(asset)

    return {
        "platform": request.platform,
        "assets": content_assets,
        "ready_to_post": True,
        "estimated_engagement": [asset.engagement_prediction for asset in content_assets]
    }

# Background processing for heavy tasks
@app.post("/ai-agent/process-video-generation")
async def process_video_generation(background_tasks: BackgroundTasks):
    """
    Handle heavy video/animation generation in background
    """
    background_tasks.add_task(generate_video_assets)
    return {"status": "processing", "check_status_url": "/ai-agent/status/{job_id}"}

```

3. N8n Sub-Workflows for Each Platform

json

```
{
  "name": "TikTok Content Creation",
  "nodes": [
    {
      "name": "AI Agent: Extract TikTok Clips",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "http://localhost:8000/ai-agent/extract-clips",
        "body": {
          "platform": "tiktok",
          "duration": "15-60",
          "style": "viral_hooks",
          "content_data": "{{json.content_analysis}}"
        }
      }
    },
    {
      "name": "Generate TikTok Video",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "https://api.runwayml.com/v1/video/generate",
        "body": {
          "prompt": "{{json.visual_concept}}",
          "duration": "{{json.clip_duration}}",
          "style": "tiktok_viral"
        }
      }
    },
    {
      "name": "Add Captions & Branding",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "http://localhost:8000/ai-agent/add-branding",
        "body": {
          "video_url": "{{json.video_url}}",
          "captions": "{{json.auto_captions}}",
          "platform": "tiktok"
        }
      }
    }
  ]
}
```



```
"name": "Schedule TikTok Post",
"type": "@n8n/nodes-base.httpRequest",
"parameters": {
  "url": "https://api.tiktok.com/v1/post/schedule",
  "body": {
    "video_url": "={{$json.final_video_url}}",
    "caption": "={{$json.optimized_caption}}",
    "schedule_time": "={{$json.optimal_time}}"
  }
}
}
```

4. AI Agent Intelligence Layer

python

```

class ContentStrategyAI:
    def __init__(self):
        self.claude_client = AnthropicClient()
        self.performance_db = PerformanceDatabase()

    async def create_platform_strategy(self, platform, content_analysis, context_insights, historical_performance):
        """
        Use Claude to create intelligent platform-specific strategies
        """

        strategy_prompt = f"""
        You are an expert social media strategist. Create a content strategy for {platform}.

        CONTENT ANALYSIS:
        {json.dumps(content_analysis, indent=2)}

        HISTORICAL CONTEXT FROM NOTEBOOKLM:
        {context_insights}

        PAST PERFORMANCE DATA:
        {json.dumps(historical_performance, indent=2)}

        PLATFORM REQUIREMENTS FOR {platform.upper()}:
        {self.get_platform_requirements(platform)}

        Generate:
        1. Top 3 content pieces to create
        2. Optimal posting times
        3. Caption strategies
        4. Visual/video concepts
        5. Hashtag recommendations
        6. Engagement predictions

        Format as JSON with specific actionable items.
        """

        response = await self.claude_client.complete(strategy_prompt)
        return json.loads(response)

    def get_platform_requirements(self, platform):
        requirements = {

```

```

'tiktok': ""
- 15-60 second videos
- Hook viewers in first 3 seconds
- Trending sounds/music
- Vertical format (9:16)
- Auto-captions essential
- Quick cuts and transitions
"",
'instagram': ""
- Mix of Reels (15-90s), Stories (15s), and Posts
- Aesthetic consistency
- Behind-the-scenes content performs well
- Story engagement through polls/questions
- Visual storytelling
"",
'linkedin': ""
- Professional insights and industry commentary
- Personal story + business lesson format
- Long-form captions (1300+ characters)
- Native video or document carousels
- Thought leadership positioning
"",
'youtube': ""
- YouTube Shorts: 15-60 seconds, vertical
- Strong thumbnails and titles
- Educational or entertaining hooks
- Clear value proposition
- Subscribe CTAs
""
}
return requirements.get(platform, "General social media best practices")

```

5. N8n Workflow Orchestration Features

javascript

```

// N8n Custom Function Node for AI Agent Communication
function aiAgentDecision() {
  const contentData = $input.all()[0].json;

  // AI Agent makes intelligent decisions about content flow
  const decisions = {
    shouldCreateVideo: contentData.viral_potential > 0.7,
    priorityPlatforms: contentData.recommended_platforms,
    urgencyLevel: contentData.trending_topics?.length > 0 ? 'high' : 'normal',
    contentType: contentData.primary_emotion === 'excitement' ? 'viral' : 'educational'
  };

  return {
    json: {
      ...contentData,
      ai_decisions: decisions,
      next_action: decisions.urgencyLevel === 'high' ? 'immediate_processing' : 'scheduled_processing'
    }
  };
}

// N8n Error Handling with AI Agent Recovery
function handleProcessingError() {
  const error = $input.all()[0].json;

  // Send error to AI Agent for intelligent recovery
  return {
    json: {
      error_type: error.type,
      recovery_action: 'ai_agent_intervention',
      alternative_approach: true,
      retry_with_different_strategy: true
    }
  };
}

```

6. Performance Feedback Loop in N8n

json

```
{
  "name": "Performance Learning Workflow",
  "trigger": "schedule",
  "interval": "daily",
  "nodes": [
    {
      "name": "Collect Platform Analytics",
      "type": "@n8n/nodes-base.merge",
      "parameters": {
        "mode": "mergeByPosition"
      }
    },
    {
      "name": "AI Agent: Analyze Performance",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "http://localhost:8000/ai-agent/analyze-performance",
        "body": {
          "analytics_data": "{{${$json.combined_analytics}}",
          "time_period": "last_24_hours"
        }
      }
    },
    {
      "name": "Update NotebookLM with Learnings",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "http://localhost:8000/ai-agent/update-knowledge",
        "body": {
          "learnings": "{{${$json.performance_insights}}",
          "successful_patterns": "{{${$json.winning_strategies}}}"
        }
      }
    },
    {
      "name": "Adjust Future Strategies",
      "type": "@n8n/nodes-base.httpRequest",
      "parameters": {
        "url": "http://localhost:8000/ai-agent/update-strategy-models",
        "body": {
          "performance_data": "{{${$json.analytics_data}}",
```

```
    "optimization_recommendations": "={{$json.optimizations}}"
  }
}
]
}
```

Implementation Strategy

Phase 1: Basic N8n + AI Agent Setup

1. Set up AI Agent API with basic content analysis
2. Create simple N8n workflow for content ingestion
3. Implement NotebookLM integration
4. Test with one platform (TikTok)

Phase 2: Multi-Platform Orchestration

1. Add platform-specific sub-workflows
2. Implement AI decision-making logic
3. Add media generation capabilities
4. Create scheduling and posting automation

Phase 3: Intelligence Layer

1. Add performance tracking and learning
2. Implement viral prediction algorithms
3. Create advanced content optimization
4. Add cross-platform content adaptation

Phase 4: Scale and Optimize

1. Add error handling and recovery
2. Implement advanced scheduling logic
3. Create analytics dashboard
4. Add multi-brand support

Key Benefits of This Architecture

Intelligent Orchestration: N8n handles the workflow complexity while AI Agent makes smart decisions

Scalable Processing: Can handle multiple content pieces simultaneously

Platform Optimization: Each platform gets AI-optimized content tailored to its audience

Learning System: Performance data feeds back to improve future content decisions

Human Oversight: Easy to add approval steps where needed

Error Recovery: AI Agent can adapt and recover from failures automatically

This setup gives you the best of both worlds - N8n's powerful workflow automation with AI Agent's intelligent decision-making!