# Tutorial 06: Multi-Agent Systems - Complex Orchestration

**Difficulty:** advanced
**Reading Time:** 1.5 hours
**Tags:** advanced, multi-agent, orchestration, complex-workflows, systems
**Description:** Build sophisticated multi-agent systems combining sequential and parallel workflows for complex business processes and decision-making systems.

# Tutorial 06: Multi-Agent Systems - Agents Working Together

## Overview

This demonstrates the fan-out/gather pattern - parallel data gathering + sequential synthesis!

## Step 1: Get the Working Implementation

A complete, tested implementation is available in the repository:

```
# Navigate to the working implementation
cd tutorial_implementation/tutorial06/

# Install dependencies
make setup

# Copy environment template and add your API key
cp content_publisher/.env.example content_publisher/.env
# Edit content_publisher/.env and add your GOOGLE_API_KEY
```

**Alternative**: Follow the step-by-step build instructions below to create your own implementation.

# Step 2: Create Project Structure (Optional - Skip if using working implementation)

If you prefer to build from scratch, create this structure:

```
mkdir content_publisher
cd content_publisher
touch __init__.py agent.py .env
```

Copy your `.env` file from previous tutorials.e art of combining Sequential and Parallel agents to build sophisticated multi-agent systems! This tutorial brings together everything you've learned to create production-ready agent architectures that can handle complex, real-world tasks.

🎯 **Working Implementation Available**: A complete, tested content publishing system is available at `tutorial_implementation/tutorial06/` (https://github.com/raphaelmansuy/adk_training/tree/main/tutorial_implementation/tutorial06). The implementation includes comprehensive tests, documentation, and a user-friendly setup process.

# Prerequisites

- **Completed Tutorials 01-05** - Understanding of agents, tools, Sequential, and Parallel patterns
- **Installed ADK** - `pip install google-adk`
- **API key configured** - From Tutorial 01
- **GoogleSearch tool access** - Available with Gemini 2.0+ models (automatically enabled)

# Core Concepts

## Multi-Agent Architecture

Real-world problems need **multiple agents working together** in sophisticated ways:

- **Sequential chains** (Tutorial 04) - Order matters: A → B → C
- **Parallel branches** (Tutorial 05) - Speed matters: A, B, C run together
- **Nested orchestration** - Combining both: Parallel inside Sequential, or vice versa
- **Specialized agents** - Each agent has ONE focused responsibility

## Common Multi-Agent Patterns

**Pattern 1: Sequential Pipeline**

```
Agent A → Agent B → Agent C
```

*Use when*: Each step needs previous step's output

**Pattern 2: Fan-Out/Gather**

```
      ┌─ Agent A ─┐
In ──┼─ Agent B ─┼─→ Merger → Out
      └─ Agent C ─┘
```

*Use when*: Gather data from multiple sources, then synthesize

**Pattern 3: Nested Workflows** (This Tutorial!)

```
        ┌─ Sequential: A → B ─┐
       In ────┼─ Sequential: C → D ─┼─→ Final Agent → Out
        └─ Sequential: E → F ─┘

     Parallel Container
```

*Use when*: Multiple independent pipelines, then final synthesis

## Real Research with GoogleSearch Tool

🎯 **Enhanced with Real Web Search**: This tutorial now uses ADK's builtin `google_search` tool to perform actual web research instead of simulated responses. This makes the content publishing system much more powerful and realistic!

The GoogleSearch tool automatically:

- Searches the web for current, relevant information
- Provides factual, up-to-date data from credible sources
- Works seamlessly with Gemini 2.0+ models
- No additional setup required beyond your API key

**Why Real Search Matters:**

- ✅ **Authentic content** - Based on actual web research
- ✅ **Current information** - Always up-to-date with latest news
- ✅ **Credible sources** - Draws from real websites and publications
- ✅ **Educational value** - Shows how to build production-ready research systems

# Use Case

We're building a **Content Publishing System** for a digital magazine that needs to:

1. **Research Phase** (3 parallel pipelines):
2. *News Pipeline*: Fetch current events → Summarize key points
3. *Social Pipeline*: Gather trending topics → Analyze sentiment

4. *Expert Pipeline*: Find expert opinions → Extract quotes
5. **Content Creation Phase** (sequential):
6. Combine all research
7. Write article draft
8. Edit for clarity
9. Format for publication

This demonstrates **nested orchestration**: 3 parallel sequential pipelines + final sequential synthesis!

# Step 1: Create Project Structure

```
mkdir content_publisher
cd content_publisher
touch __init__.py agent.py .env
```

Copy your `.env` file from previous tutorials.

# Step 2: Set Up Package Import

**content_publisher/**init**.py**

```
from . import agent
```

# Step 3: Build the Multi-Agent System

**content_publisher/agent.py**

```python
from __future__ import annotations

from google.adk.agents import Agent, ParallelAgent, SequentialAgent
from google.adk.tools import google_search

# ========================================================
# PARALLEL BRANCH 1: News Research Pipeline
# ========================================================
news_fetcher = Agent(
    name="news_fetcher",
    model="gemini-2.0-flash",
    description="Fetches current news articles using Google Search",
    instruction=(
        "You are a news researcher. Based on the user's topic, search for "
        "current news articles and recent developments.\n"
        "\n"
        "Use the google_search tool to find 3-4 current news articles.\n"
        "Focus on recent, credible news sources from the past 6 months.\n"
        "\n"
        "Output a bulleted list with:\n"
        "• Source + Headline + Brief summary\n"
        "• Include publication dates when available\n"
        "\n"
        "Search query should be: '[topic] news recent developments site:reputa
    ),
    tools=[google_search],
    output_key="raw_news"
)

news_summarizer = Agent(
    name="news_summarizer",
    model="gemini-2.0-flash",
    description="Summarizes key news points",
    instruction=(
        "Summarize the news articles into 2-3 key takeaways.\n"
        "\n"
        "**Raw News:**\n"
        "{raw_news}\n"
        "\n"
        "Output format:\n"
        "KEY TAKEAWAYS:\n"
        "1. First key point\n"
        "2. Second key point\n"
        "3. Third key point"
    ),
    output_key="news_summary"
```

```
)

# Sequential pipeline for news research
news_pipeline = SequentialAgent(
    name="NewsPipeline",
    sub_agents=[news_fetcher, news_summarizer],
    description="Fetches and summarizes news"
)

# ========================================================
# PARALLEL BRANCH 2: Social Media Research Pipeline
# ========================================================
social_monitor = Agent(
    name="social_monitor",
    model="gemini-2.0-flash",
    description="Monitors social media trends using Google Search",
    instruction=(
        "You are a social media analyst. Based on the user's topic, search for
        "trending discussions, popular hashtags, and public sentiment.\n"
        "\n"
        "Use the google_search tool to find:\n"
        "• Trending hashtags and topics on social platforms\n"
        "• Recent social media discussions and viral content\n"
        "• Public opinion and sentiment analysis\n"
        "\n"
        "Search for: '[topic] social media trends reddit twitter discussion'\n
        "\n"
        "Output:\n"
        "• 3-4 trending hashtags or topics\n"
        "• Popular discussion themes\n"
        "• General sentiment (positive/negative/mixed) with evidence"
    ),
    tools=[google_search],
    output_key="raw_social"
)

sentiment_analyzer = Agent(
    name="sentiment_analyzer",
    model="gemini-2.0-flash",
    description="Analyzes social sentiment",
    instruction=(
        "Analyze the social media data and extract key insights.\n"
        "\n"
        "**Social Media Data:**\n"
        "{raw_social}\n"
        "\n"
        "Output format:\n"
```

```
        "SOCIAL INSIGHTS:\n"
        "• Trending: [hashtags/topics]\n"
        "• Sentiment: [overall mood]\n"
        "• Key Themes: [main discussion points]"
    ),
    output_key="social_insights"
)

# Sequential pipeline for social research
social_pipeline = SequentialAgent(
    name="SocialPipeline",
    sub_agents=[social_monitor, sentiment_analyzer],
    description="Monitors and analyzes social media"
)

# ======================================================
# PARALLEL BRANCH 3: Expert Opinion Pipeline
# ======================================================
expert_finder = Agent(
    name="expert_finder",
    model="gemini-2.0-flash",
    description="Finds expert opinions using Google Search",
    instruction=(
        "You are an expert opinion researcher. Based on the user's topic, sear
        "what industry experts, academics, or thought leaders are saying.\n"
        "\n"
        "Use the google_search tool to find:\n"
        "• Industry experts and their credentials\n"
        "• Academic researchers and their affiliations\n"
        "• Thought leaders and their recent statements\n"
        "\n"
        "Search for: '[topic] expert opinion academic research thought leader'
        "\n"
        "Output:\n"
        "• 2-3 expert names and their credentials\n"
        "• Their key statements or positions\n"
        "• Source (where they said it) with links when available"
    ),
    tools=[google_search],
    output_key="raw_experts"
)

quote_extractor = Agent(
    name="quote_extractor",
    model="gemini-2.0-flash",
    description="Extracts quotable insights",
    instruction=(
```

```
        "Extract the most impactful quotes and insights from expert opinions.\
        "\n"
        "**Expert Opinions:**\n"
        "{raw_experts}\n"
        "\n"
        "Output format:\n"
        "EXPERT INSIGHTS:\n"
        "• Quote 1: \"...\" - [Expert Name], [Credentials]\n"
        "• Quote 2: \"...\" - [Expert Name], [Credentials]"
    ),
    output_key="expert_quotes"
)

# Sequential pipeline for expert research
expert_pipeline = SequentialAgent(
    name="ExpertPipeline",
    sub_agents=[expert_finder, quote_extractor],
    description="Finds and extracts expert opinions"
)

# ========================================================
# PHASE 1: PARALLEL RESEARCH (3 pipelines run together!)
# ========================================================
parallel_research = ParallelAgent(
    name="ParallelResearch",
    sub_agents=[
        news_pipeline,    # Sequential: fetch → summarize
        social_pipeline,  # Sequential: monitor → analyze
        expert_pipeline   # Sequential: find → extract
    ],
    description="Runs all research pipelines concurrently"
)

# ========================================================
# PHASE 2: CONTENT CREATION (Sequential synthesis)
# ========================================================
article_writer = Agent(
    name="article_writer",
    model="gemini-2.0-flash",
    description="Writes article draft from all research",
    instruction=(
        "You are a professional writer. Write an engaging article using ALL "
        "the research below.\n"
        "\n"
        "**News Summary:**\n"
        "{news_summary}\n"
        "\n"
```

```
        "**Social Insights:**\n"
        "{social_insights}\n"
        "\n"
        "**Expert Quotes:**\n"
        "{expert_quotes}\n"
        "\n"
        "Write a 4-5 paragraph article that:\n"
        "- Opens with a compelling hook\n"
        "- Incorporates news, social trends, and expert opinions naturally\n"
        "- Uses expert quotes effectively\n"
        "- Has a strong conclusion\n"
        "\n"
        "Output ONLY the article text."
    ),
    output_key="draft_article"
)

article_editor = Agent(
    name="article_editor",
    model="gemini-2.0-flash",
    description="Edits article for clarity and impact",
    instruction=(
        "You are an editor. Review and improve the article below.\n"
        "\n"
        "**Draft Article:**\n"
        "{draft_article}\n"
        "\n"
        "Edit for:\n"
        "- Clarity and flow\n"
        "- Impact and engagement\n"
        "- Grammar and style\n"
        "\n"
        "Output the improved article."
    ),
    output_key="edited_article"
)

article_formatter = Agent(
    name="article_formatter",
    model="gemini-2.0-flash",
    description="Formats article for publication",
    instruction=(
        "Format the article for publication with proper markdown.\n"
        "\n"
        "**Article:**\n"
        "{edited_article}\n"
        "\n"
```

```
        "Add:\n"
        "- Compelling title (# heading)\n"
        "- Byline (By: AI Content Team)\n"
        "- Section headings where appropriate (## subheadings)\n"
        "- Proper formatting (bold, italic, quotes)\n"
        "- Publication date placeholder\n"
        "\n"
        "Output the final formatted article."
    ),
    output_key="published_article"
)

# ========================================================
# COMPLETE MULTI-AGENT SYSTEM
# ========================================================
content_publishing_system = SequentialAgent(
    name="ContentPublishingSystem",
    sub_agents=[
        parallel_research,  # Phase 1: Research (3 parallel pipelines!)
        article_writer,     # Phase 2: Draft
        article_editor,     # Phase 3: Edit
        article_formatter   # Phase 4: Format
    ],
    description="Complete content publishing system with parallel research and
)

# MUST be named root_agent for ADK
root_agent = content_publishing_system
```

## Architecture Visualization

```
User: "Write article about electric vehicles"
    ↓
┌──────────────────────────────────────────────────────────┐
|  PHASE 1: Parallel Research (3 Sequential Pipelines)     |
├──────────────────────────────────────────────────────────┤
|  News Pipeline:    fetch → summarize → news_summary      |
|  Social Pipeline:  monitor → analyze → social_insights   | ← ALL RUN
|  Expert Pipeline:  find → extract → expert_quotes        |   AT ONCE!
└──────────────────────────────────────────────────────────┘
    ↓ (waits for ALL 3 to complete)
┌──────────────────────────────────────────────────────────┐
|  PHASE 2: Sequential Content Creation                    |
├──────────────────────────────────────────────────────────┤
|  Writer:    combines all research → draft_article        |
|  Editor:    reviews draft → edited_article               | ← ONE AT
|  Formatter: adds markdown → published_article            |   A TIME
└──────────────────────────────────────────────────────────┘
    ↓
Final Output: Publication-ready article!
```

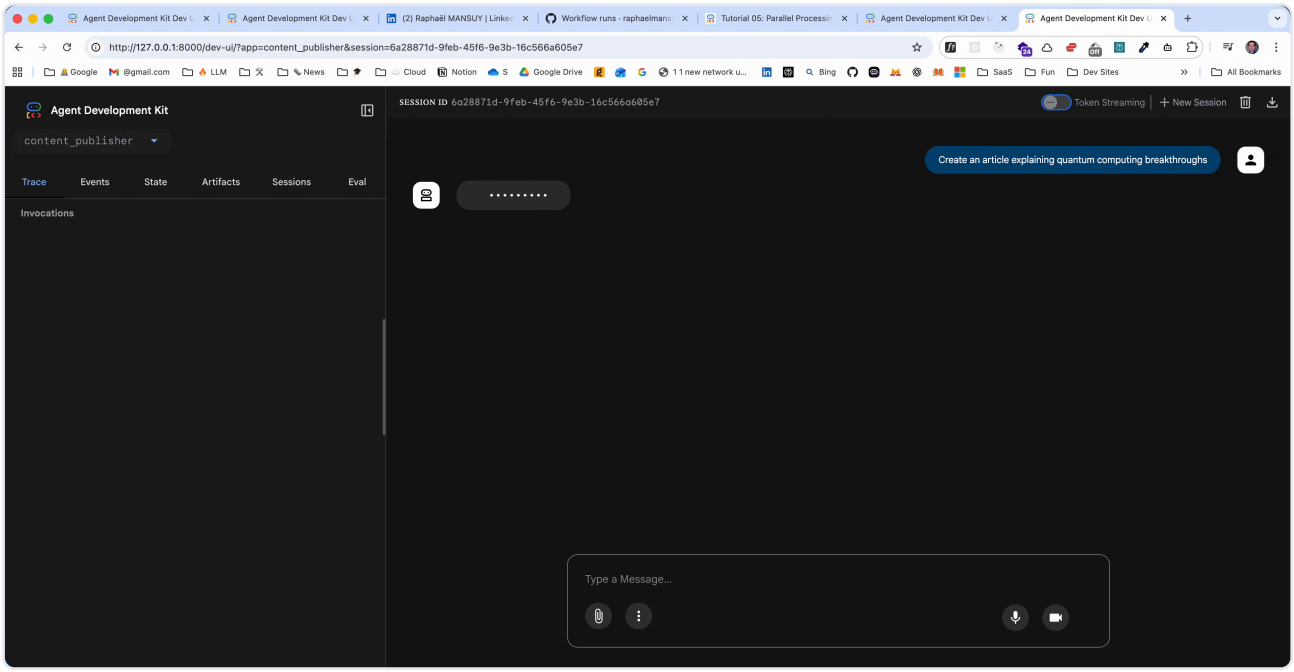
**Why This Architecture Works:**

1. **Phase 1 (Parallel Research)**:
2. 3 independent research areas (news, social, expert)
3. Each has 2-step pipeline (gather → process)
4. All 3 pipelines run simultaneously = FAST
5. **Phase 2 (Sequential Creation)**:
6. Writer NEEDS all 3 research outputs
7. Editor NEEDS writer's draft
8. Formatter NEEDS editor's version
9. Must run in order = QUALITY
10. **Best of Both Worlds**:
11. Speed from parallelism (research phase)
12. Quality from sequential flow (creation phase)

# Step 4: Run the Publishing System

## Demo in Action

Here's what your multi-agent publishing system looks like in action:




## Using the Working Implementation

```
# From tutorial_implementation/tutorial06/
make dev
```

Open `http://localhost:8000` and select "content_publisher".

## Manual Setup (if building from scratch)

Navigate to parent directory and launch:

```
cd ..  # Go to parent of content_publisher/
adk web
```

Open `http://localhost:8000` and select "content_publisher".

## Try These Prompts

**Technology Topic:**

```
Write an article about artificial intelligence in healthcare
```

**Current Events:**

```
Create an article about renewable energy adoption
```

**Business Topic:**

```
Write about the future of remote work
```

**Science Topic:**

```
Create an article explaining quantum computing breakthroughs
```

# Understanding the Execution Flow

Open the **Events tab** to see the sophisticated orchestration:

**Phase 1: Parallel Research Starts**

1. Event: ParallelResearch starts
2. Events (ALL SIMULTANEOUSLY):
3. NewsPipeline starts
    1. news_fetcher runs
    2. news_summarizer runs
4. SocialPipeline starts
    1. social_monitor runs
    2. sentiment_analyzer runs
5. ExpertPipeline starts
    1. expert_finder runs

2. quote_extractor runs
6. Event: ParallelResearch completes (when ALL 3 pipelines done)

**Phase 2: Sequential Content Creation** 4. Event: article_writer starts (with all 3 research outputs injected) 5. Event: article_writer completes 6. Event: article_editor starts (with draft) 7. Event: article_editor completes 8. Event: article_formatter starts (with edited version) 9. Event: article_formatter completes → DONE!

Watch how 6 agents in Phase 1 run concurrently, then 3 agents in Phase 2 run sequentially!

# Testing Your Implementation

The working implementation includes comprehensive tests to validate your understanding:

```
# From tutorial_implementation/tutorial06/
make test
```

**Test Coverage:**

- ✅ Individual agent configurations (9 agents)
- ✅ Sequential pipeline structures (3 pipelines)
- ✅ Parallel research orchestration
- ✅ Complete system integration
- ✅ State management and data flow
- ✅ Import validation and module structure
- ✅ Project file organization

**Quick Demo:**

```
# Test basic functionality without full ADK setup
make demo
```

This validates that your agents load correctly and the pipeline structure is sound.

# Expected Behavior

**Example: "Write an article about electric vehicles"**

```
User: Write an article about electric vehicles

[Phase 1: All 3 pipelines run in parallel - watch Events!]

News Summary:
KEY TAKEAWAYS:
1. EV sales hit record 1.2M units in Q3 2024 (Source: Bloomberg, Oct 2024)
2. Tesla Cybertruck production begins with 1,500 deliveries (Reuters, Oct 2024
3. EU mandates all new cars to be zero-emission by 2035 (BBC News, Sep 2024)

Social Insights:
SOCIAL INSIGHTS:
● Trending: #ElectricVehicles #EVs #SustainableTransport #Tesla
● Sentiment: 78% positive, growing adoption excitement
● Key Themes: Range anxiety decreasing, charging infrastructure improving, cos

Expert Quotes:
EXPERT INSIGHTS:
● "Battery costs have dropped 90% since 2010, making EVs cheaper to own" - Dr.
● "By 2030, EVs will represent 60% of new vehicle sales globally" - BloombergN
● "The transition to electric vehicles is accelerating faster than predicted"

[Phase 2: Sequential content creation]

[Writer combines all research into draft]
[Editor improves clarity and flow]
[Formatter adds publication styling]

Final Output:
# The Electric Vehicle Revolution: Accelerating Toward a Sustainable Future

**By: AI Content Team** | *October 2024*

The automotive industry is undergoing its most dramatic transformation since t

[Complete formatted article with real research data, expert quotes, and social
```

**Note**: Actual search results will vary based on current events and web content at the time of execution. The system now performs real Google searches for authentic, up-to-date information!

# How It Works (Behind the Scenes)

**Nested Agent Execution:**

1. **Outer SequentialAgent** controls main phases:
2. sub_agents = [parallel_research, writer, editor, formatter]
3. **parallel_research (ParallelAgent)** runs 3 sub-agents concurrently:
4. Each sub-agent is itself a SequentialAgent!
5. NewsPipeline runs: fetcher → summarizer
6. SocialPipeline runs: monitor → analyzer
7. ExpertPipeline runs: finder → extractor
8. **State Management**:
9. Each pipeline saves to its own output_key
10. Writer reads `{news_summary}`, `{social_insights}`, `{expert_quotes}`
11. Editor reads `{draft_article}`
12. Formatter reads `{edited_article}`

**Performance Characteristics:**

- **Without parallelism**: ~60 seconds (6 research agents + 3 creation agents)
- **With parallelism**: ~25 seconds (6 research agents run together!)
- **Speedup**: ~2.4x faster

# Key Takeaways

✅ **Nest Sequential inside Parallel** - Multiple independent pipelines running concurrently

✅ **Nest Parallel inside Sequential** - Phases of work where one phase needs parallelism

✅ **Each agent has ONE job** - Specialized, focused, testable

✅ **State flows through output_keys** - Explicit data dependencies

✅ **Sophisticated orchestration is simple** - Just compose Sequential + Parallel

✅ **Real performance gains** - Parallel research phase is 3x faster

✔️ **Production-ready patterns** - Used in real content systems, data pipelines, analysis tools

# Best Practices

**DO:**

- Draw your architecture before coding (visualize the flow!)
- Keep agents focused (one clear responsibility)
- Use descriptive names (news_fetcher, not agent1)
- Plan state flow (what keys does each agent need?)
- Test individual pipelines before composing
- Monitor Events tab to verify execution

**DON'T:**

- Over-nest (3+ levels deep gets confusing)
- Create agents that do multiple things
- Forget to set output_keys (breaks state flow!)
- Assume execution order in parallel blocks
- Skip the planning phase (complexity needs design!)

# Common Issues

**Problem**: "Agents in parallel block seem to run sequentially"

- **Solution**: Check Events tab for actual start times
- **Solution**: Model API might be rate-limiting concurrent requests

**Problem**: "Writer agent missing research data"

- **Solution**: Verify each pipeline sets its output_key
- **Solution**: Check `{key}` names in writer instruction match exactly

**Problem**: "One research pipeline fails, whole system stops"

- **Solution**: ParallelAgent waits for ALL - one failure blocks

- **Solution**: Add error handling or retry logic to individual agents

**Problem**: "Hard to debug nested agents"

- **Solution**: Test each pipeline individually first
- **Solution**: Use Events tab to trace execution flow
- **Solution**: Add descriptive agent names and descriptions

# What We Built

You now have a production-grade content publishing system that:

- Researches from 3 independent sources concurrently
- Processes each source with specialized pipelines
- Synthesizes everything into publication-ready content
- Demonstrates sophisticated multi-agent orchestration

And you understand how to architect complex agent systems!

# Real-World Applications

**Multi-Agent Systems Are Perfect For:**

- **Content Platforms**: Research → Write → Edit → Publish (this tutorial!)
- **Data Analysis**: Gather data (parallel) → Merge → Analyze → Visualize
- **E-Commerce**: Check inventory + pricing + reviews (parallel) → Recommend
- **Customer Support**: Classify → Route (parallel specialists) → Respond → Follow-up
- **Financial Analysis**: Market data + news + sentiment (parallel) → Analyze → Report
- **Code Generation**: Design + Implement + Review + Test + Document

# Next Steps

🚀 **Tutorial 07: Loop Agents** - Learn iterative refinement for quality improvement

📖 **Further Reading**:

- Workflow Agents Overview (https://google.github.io/adk-docs/agents/workflow-agents/)
- Agent Composition Patterns (https://google.github.io/adk-docs/agents/composition/)
- State Management (https://google.github.io/adk-docs/sessions/state/)

# Exercises (Try On Your Own!)

1. **Add more research pipelines** - Academic papers, patent databases, competitor analysis
2. **Conditional routing** - Have a classifier decide which pipelines to run
3. **Quality control** - Add a fact-checker agent before publication
4. **A/B testing** - Run 2 parallel writer agents, have editor choose best
5. **Nested parallel** - Make each research pipeline spawn its own parallel sub-agents

# Complete Code Reference

**Working Implementation**: See `tutorial_implementation/tutorial06/` (https://github.com/raphaelmansuy/adk_training/tree/main/tutorial_implementation/tutorial06) for a complete, tested version with comprehensive documentation.

**Key Files:**

- `content_publisher/agent.py` (https://github.com/raphaelmansuy/adk_training/blob/main/tutorial_implementation/tutorial06/content_publisher/agent.py) - Complete multi-agent orchestration
- `tests/test_agent.py` (https://github.com/raphaelmansuy/adk_training/blob/main/tutorial_implementation/tutorial06/tests/test_agent.py) - 62 comprehensive tests
- `README.md` (https://github.com/raphaelmansuy/adk_training/blob/main/tutorial_implementation/tutorial06/README.md) - Detailed implementation guide
- `Makefile` (https://github.com/raphaelmansuy/adk_training/blob/main/tutorial_implementation/tutorial06/Makefile) - Development commands

**Quick Start with Working Code:**

```
cd tutorial_implementation/tutorial06/
make setup  # Install dependencies
make test   # Run all tests (62 passing)
make dev    # Start development server
```

**Manual Implementation:**

**content_publisher/**init**.py**

```
from . import agent
```

**content_publisher/.env**

```
GOOGLE_GENAI_USE_VERTEXAI=FALSE
GOOGLE_API_KEY=your-api-key-here
```

**content_publisher/agent.py**

```
# See Step 3 above for complete code
```

Congratulations! You've mastered multi-agent orchestration! 🎯🚀📰

---

Generated on 2025-10-19 17:56:21 from 06_multi_agent_systems.md

Source: Google ADK Training Hub