Autonomous Content Ecosystems: Technical Architecture, Market Strategy, and Code Implementation for Faceless Video Automation

1. Introduction: The Evolution of Algorithmic Media

The digital content landscape is undergoing a paradigm shift comparable to the industrial revolution's transition from artisanal craft to mass production. We are witnessing the dawn of "Autonomous Content Ecosystems"—systems where the entire lifecycle of media production, from ideation to distribution, is managed by software agents rather than human creators. This report provides a comprehensive blueprint for constructing such a system: a fully local, zero-cost, and automated pipeline for generating high-retention short-form video content for platforms like YouTube Shorts, Instagram Reels, and TikTok.

The objective is not merely automation, but *verification*. The historical weakness of automated content has been quality control—generating "spam" that algorithms penalize. By implementing a "Critic-Refiner" architecture using local Large Language Models (LLMs), we introduce a self-regulating mechanism that mimics the editorial oversight of a human production team. This report dissects the technical implementation of this architecture using n8n as the orchestrator, Ollama for cognition, ComfyUI for visual synthesis, and Python for assembly, all while adhering to a strict "no-cloud-API" constraint to ensure data privacy and zero marginal cost.

Furthermore, we anchor this technical exploration in a rigorous market analysis. We evaluate the current "faceless" content economy, identifying unsaturated niches with high Revenue Per Mille (RPM) potential, and analyze existing open-source repositories—specifically auto-market-pulse, MoneyPrinterTurbo, and ShortGPT—to extract reusable code logic for rapid development.

---

2. Foundational Analysis: From Market Pulse to Content Pulse

2.1 Architectural Forensics: The auto-market-pulse Repository

The user’s starting point, the auto-market-pulse repository , serves as more than just a code reference; it establishes the fundamental logic required for automated decision-making. While

the repository itself is an Android application designed for financial market tracking, its architectural patterns—specifically the handling of asynchronous data streams and criteria-based triggering—are directly homologous to the requirements of an autonomous video engine.

2.1.1 Structural Decomposition

The auto-market-pulse repository utilizes a classic Model-View-ViewModel (MVVM) architecture, leveraging LiveData and OkHttp for network operations.

- **Data Ingestion (OkHttp):** The app polls a server for market data. In our video context, this translates to the "Trend Acquisition" layer, where python scripts (using pytrends or scraping libraries) replace the financial API endpoints to poll Google Trends or Reddit for viral signals.
- **Criteria Evaluation (GSON/Parsers):** The repository parses JSON responses to check if specific market conditions (criteria) are met. This logic is the precursor to our "Critic" agent. Instead of checking if stock_price > threshold, our agent will check if trend_velocity > threshold or if a script's virality_score > 8.
- **Reactive State (LiveData):** The app updates the UI based on data changes. In an automated pipeline, this "UI update" is replaced by the *triggering of the next workflow stage* within n8n. The "pulse" becomes the heartbeat of the content scheduler.

2.1.2 The Shift to Server-Side Orchestration

Transitioning this logic from a mobile app (client-side) to a video automation server involves a shift in technology stack but preservation of the *event-driven* philosophy.

- **Java/Kotlin -> Python/Node.js:** The logic encapsulated in auto-market-pulse's Java classes for parsing criteria must be reimplemented in Python nodes within n8n.
- **Monolith -> Microservices:** Instead of a single app, the functionality is distributed across Docker containers (n8n, Ollama, ComfyUI). The OkHttp calls are replaced by n8n's HTTP Request nodes, communicating over the Docker bridge network.

2.2 The "Market Pulse" of 2026: Content Velocity

Just as financial markets have "tickers," the attention economy has "keywords." The first step in our pipeline is not generation, but *listening*. We repurpose the concept of a "Market Pulse" to build a "Content Pulse" engine. This engine continuously scans the digital horizon for topics that are:

1. **Rising:** High velocity in search volume (Google Trends).
2. **Unsaturated:** High demand but low supply of quality video (Keyword competitiveness).
3. **Visualizable:** Topics that can be represented by AI imagery (avoiding abstract concepts that confuse diffusion models).

This foundational understanding—that automation must be *driven by data*, not just random triggers—sets the stage for our market research. We do not want to generate content into a

void; we want to generate into a demand spike.

---

3. Market Research: The Faceless Economy in 2026

The "faceless" content market—videos produced without on-camera talent—has matured significantly. In the early 2020s, simple arbitrage (e.g., reposting podcasts, generic motivational quotes) was viable. By 2026, platform algorithms have evolved to detect and suppress low-effort, repetitive content. Success now requires "High-Signal" content: videos that deliver dense value, unique aesthetics, or compelling narrative structures.

3.1 The Saturation Spectrum

To identify the "easy to implement" yet "viral" ideas requested, we must categorize niches based on their saturation levels and revenue potential (RPM).

3.1.1 The Red Ocean (High Saturation, Avoid)

Data from 2024-2025 indicates that certain niches have reached a saturation point where new entrants face near-impossible odds without massive differentiation.[2]

- **Generic Motivation:** The "grindset" niche is flooded. Simple text-over-video quotes no longer generate significant reach unless accompanied by original, high-quality cinematic visuals or unique audio engineering.
- **Standard Rain/Lofi:** The barrier to entry was too low. Millions of channels pump out "Rain Sounds for Sleep." Without specific sub-niching (e.g., "ASMR 1920s Train Cabin"), growth is stagnant.
- **Reddit Readings (TTS):** The era of reading r/AskReddit threads with robotic voices is over. Platforms now de-monetize this as "reused content."

3.1.2 The Blue Ocean (High RPM, Low Saturation)

The following niches represent the optimal targets for an automated system in 2026. They are characterized by "Information Asymmetry"—the viewer wants to know something specific, and the video delivers it efficiently.

Table 1: Strategic Niche Analysis for Automated Faceless Channels

Niche Category	Sub-Niche Concept	Viral Driver	Automation Complexity	RPM Potential
Visual	**Data**	Visualizing invisible forces	Medium (Python	**$15 - $50** (Finance

Economics	**Storytelling**	(inflation, debt) satisfies anxiety & curiosity.	charting + AI Backgrounds)	advertisers) [2]
Micro-History	**"The Weird Past"**	"Did you know?" hooks trigger immediate retention. Focus on hygiene, food, or daily life of eras.	Low (ComfyUI Vintage Style + Edge-TTS)	**$6 - $9** (Broad appeal) [5]
Future Tech	**AI Tool Reviews**	"Evergreen" novelty. There is always a new tool. High B2B value.	Medium (Screen recording or AI Mockups)	**$12 - $22** (Tech/SaaS advertisers) [5]
Productivity	**"Digital Nomad" Aesthetic**	Sells a fantasy lifestyle. "Dream Desk Setups" (AI Generated).	Low (ComfyUI Interior Design + Script)	**$8 - $15** (Software Affiliates) [6]
Philosophy	**Stoic Narratives**	Stories over quotes. "The Parable of the Horse." Dark, cinematic "Statue" aesthetic.	Low (Static AI Art + Ken Burns Effect)	**$4 - $8** (High Retention) [3]

3.2 Deep Dive: The "Visual Economics" Opportunity

This niche stands out as the highest ROI target. Viewers crave understanding of complex economic systems, but traditional news is boring.

- **The Format:** 60-second vertical video.
- **The Visuals:** Animated line charts (generated via Python matplotlib or manim) overlaid

on "Cinematic, dark mode, neon" backgrounds generated by ComfyUI.
- **The Script:** "Why your dollar is dying." (Fear/Greed hooks).
- **Why It Works for Automation:** Data is structured. You can feed a CSV of inflation rates into the system, and it outputs a video. No "creative" writing needed, just data interpretation.[4]

3.3 Deep Dive: "Micro-History" & The Curiosity Gap

Instead of "The Civil War" (too broad), the automated channel focuses on "What did Civil War soldiers eat?"

- **The Visuals:** ComfyUI is perfect here. "Hyper-realistic oil painting of hardtack biscuits, 1860s style."
- **The Script:** Fact-based, dense, rapid-fire.
- **Saturation:** Low. Most history channels are long-form documentaries. The "Shorts" market for history is under-served by high-quality visuals.[7]

3.4 The "Viral Formula" Requirement

Regardless of the niche, the "Critic" agent we build must enforce a strict structural formula to guarantee virality.

1. **The Hook (0-3s):** Pattern Interrupt. Visual or auditory spike.
2. **The Retention (3-15s):** The "Meat." Immediate value delivery. No intros.
3. **The Loop (End):** The final sentence must syntactically connect to the first sentence to encourage re-watching.[8]

---

4. Technical Architecture: The Local Stack

To adhere to the user's requirement for "full local and free data," we must reject the standard SaaS stack (OpenAI, ElevenLabs, Zapier) in favor of a self-hosted, containerized architecture. This approach not only eliminates monthly costs but ensures complete data privacy and sovereignty.

4.1 The Infrastructure: Docker-Based Microservices

The system runs on a single host machine (Linux/Windows WSL2) with a capable NVIDIA GPU (12GB+ VRAM recommended). We utilize **Docker** to encapsulate each component, creating a private "intranet" of AI services.

Table 2: The Local AI Stack

Service	Component	Function	Port	Local Alternative to

Orchestrator	n8n	Workflow engine. Manages logic, loops, and script execution.	5678	Zapier / Make
Cognition	Ollama	Runs local LLMs (Llama-3, Mistral) for scripting and critique.	11434	OpenAI GPT-4
Vision	ComfyUI	Generates images/videos using Stable Diffusion (SDXL/Flux).	8188	Midjourney / DALL-E
Audio	Edge-TTS	Generates neural speech from text. (Ran via CLI).	N/A	ElevenLabs
Assembly	Python	Stitches assets using FFmpeg/MoviePy.	N/A	Premiere Pro

4.2 The Orchestrator: n8n

n8n is the central nervous system. Unlike Zapier, it can be self-hosted and supports complex execution flows like loops, error handling, and local shell execution.[9]

- **Configuration:** The n8n container must have access to a shared volume (e.g., /local_files) where it can read/write assets generated by the other containers.
- **Network:** It communicates with ollama and comfyui via HTTP requests (e.g., http://ollama:11434/api/generate).

4.3 The "Self-Verification" Mechanism (Constitutional AI)

The most critical innovation in this architecture is the **Critic-Refiner Loop**. Most automated channels fail because they publish "Draft 1." Our system implements a recursive quality control process.

The Workflow Logic:

1. **Generation Phase:** Agent A (The Writer) drafts a script based on the daily trend.
2. **Verification Phase:** Agent B (The Critic) analyzes the script against a specific "Viral Rubric" (See Section 5.1). It outputs a score (0-10) and specific feedback.
3. **Decision Gate:** An If node in n8n checks the score.
    - **Fail (< 8/10):** The feedback is passed to Agent C (The Refiner), who rewrites the script. The loop repeats.
    - **Pass (>= 8/10):** The script moves to the production queue.
4. **Safety Valve:** A "Max Iterations" counter prevents infinite loops if the model cannot satisfy the critic.[10]

---

5. The Cognitive Layer: Ollama & Prompt Engineering

The quality of the output is entirely dependent on the quality of the prompts. Since we are using local models (which may be less intelligent than GPT-4), we must use "Chain of Thought" prompting and strict structural constraints.

5.1 The "Critic" Agent Prompt

The Critic is not asked to "improve" the script; it is asked to *destroy* it. This adversarial relationship forces higher quality.

System Prompt (JSON Output Required):

> "You are a ruthlessly analytical viral content strategist for TikTok. Your job is to critique video scripts. You do not write; you only grade.
>
> **The Viral Rubric:**
>
> 1. **Hook Velocity (0-10):** Does the first sentence contain a 'Pattern Interrupt' or 'Curiosity Gap'? If it starts with 'Hello' or 'Today', score is 0.
> 2. **Rhythm & Pacing (0-10):** Are sentences short (under 12 words)? Is there a variety in sentence length?
> 3. **Visualizability (0-10):** Can this script be easily matched with stock or AI images? Abstract concepts score low.
> 4. **Loop Factor (0-10):** Does the final sentence syntactically lead back into the first sentence?
>
> Output Format:
> Return ONLY a JSON object: { "hook_score": int, "rhythm_score": int,

"visual_score": int, "loop_score": int, "average_score": float, "critique": "string", "pass": boolean }. Pass is true ONLY if average_score > 8."

5.2 The "Writer" Agent Prompt

The Writer is given a "Persona" to prevent generic AI tone.

System Prompt:

"You are a cynical, fast-talking historian who loves obscure facts. You are writing a script for a 60-second YouTube Short.
Topic: {{ $json.trend }}
Constraints:

- No intros. Start immediately with the fact.
- Use active voice only.
- Maximum 140 words.
- Tone: Conspiratorial, excited, fast.
- End with a sentence that connects to the start."

---

6. The Visual Engine: ComfyUI & AnimateDiff

Static images are boring; full video generation is slow and inconsistent. The middle ground is **AnimateDiff**—a stable diffusion extension that adds motion to generated images.[12]

6.1 Why ComfyUI?

ComfyUI is node-based, making it the visual equivalent of n8n. It exposes an API where the entire workflow graph is passed as a JSON object. This allows n8n to dynamically inject prompts into specific nodes within the visual pipeline.[14]

6.2 The JSON Workflow Strategy

To automate ComfyUI, we design a workflow in the GUI, export it as JSON, and then templatize it in n8n.

The Workflow Structure:

1. **Load Checkpoint:** Juggernaut XL (Photorealism) or DreamShaper (Stylized).
2. **CLIP Text Encode (Prompt):** n8n injects: *"Cinematic shot, {{ $json.scene_description }}, 8k, dramatic lighting, highly detailed"*.
3. **AnimateDiff Loader:** Loads the motion module (mm_sd_v15_v2.ckpt). This introduces camera movement (pan, zoom) and subtle subject animation.[13]
4. **KSampler:** Generates a 16-frame GIF (approx. 2 seconds).
5. **Video Combine:** Stitches frames into an .mp4 segment.

6. **Save Image:** Outputs the file to the shared /local_files volume.

Optimization:
For speed, we use SDXL Turbo or LCM (Latent Consistency Models) which can generate quality images in 4-8 steps instead of 20-30. This reduces generation time from ~30s per image to ~5s per image.17

---

7. Code Implementation: The "Frankenstein" Assembler

The user requested to "scan github public repos" and "fork and implement parts." We have identified two key repositories: **MoneyPrinterTurbo** (harry0703/MoneyPrinterTurbo) and **ShortGPT** (RayVentura/ShortGPT). We will not simply clone them; we will surgically extract their best components to build a custom assemble_video.py script.

7.1 Component Extraction Strategy

From MoneyPrinterTurbo [18]:

- **The Gem:** The subtitle module. Aligning text to speech (Word-Level Timestamps) is mathematically complex. MoneyPrinterTurbo uses faster_whisper or edge-tts metadata to generate karaoke-style subtitles where the current word is highlighted.
- **Implementation:** We copy the app/services/subtitle.py logic. It handles the parsing of timestamp data into a format MoviePy can understand.

From ShortGPT [20]:

- **The Gem:** The EdgeTTS wrapper and Asset Sourcing. ShortGPT has a clean implementation for managing Edge-TTS voice selection and rate limiting.
- **Implementation:** We copy the EdgeTTSVoiceModule class. It simplifies the CLI interaction with the TTS engine.

7.2 The Custom Stitching Script (assemble.py)

This script acts as the "Editor." It takes the assets generated by the previous steps and compiles the final video.

Python

```
import sys
```

```python
import os
import asyncio
import edge_tts
from moviepy.editor import *
from moviepy.video.tools.subtitles import SubtitlesClip

# --- PART 1: The Audio Engine (Adapted from ShortGPT) ---
async def generate_voiceover(text, output_file, voice="en-US-ChristopherNeural"):
    """
    Uses Edge-TTS (Free) to generate audio.
    'ChristopherNeural' is chosen for its documentary-style gravity.
    """
    communicate = edge_tts.Communicate(text, voice)
    await communicate.save(output_file)

# --- PART 2: The Subtitle Engine (Adapted from MoneyPrinterTurbo) ---
def create_fancy_subtitles(text_clips_data):
    """
    Generates TextClips with specific styling (The 'Hormozi' style).
    Yellow font, black stroke, rapid popping.
    """
    clips =
    for data in text_clips_data:
        # logic to create TextClip with moviepy
        # font="Impact", fontsize=70, color='yellow', stroke_color='black'
        pass
    return clips

# --- PART 3: The Assembly Logic ---
def assemble_video(script_text, visual_assets_dir):
    # 1. Audio
    asyncio.run(generate_voiceover(script_text, "audio.mp3"))
    audio_clip = AudioFileClip("audio.mp3")

    # 2. Visuals (ComfyUI Assets)
    visual_files = sorted([f for f in os.listdir(visual_assets_dir) if f.endswith('.mp4')])
    visual_clips =

    # Calculate duration per scene based on audio length / num_scenes
    scene_duration = audio_clip.duration / len(visual_files)

    for vid_file in visual_files:
        clip = VideoFileClip(os.path.join(visual_assets_dir, vid_file))
        # Loop or speed up/slow down to match scene_duration
```

```
        clip = clip.loop(duration=scene_duration)
        visual_clips.append(clip)

    final_video = concatenate_videoclips(visual_clips, method="compose")
    final_video = final_video.set_audio(audio_clip)

    # 3. Subtitles Overlay
    # (Implementation of timestamp alignment logic derived from MoneyPrinterTurbo)
    # final_video = CompositeVideoClip([final_video, subtitle_clips])

    # 4. Render
    final_video.write_videofile("final_output.mp4", fps=30, codec="libx264", audio_codec="aac")

if __name__ == "__main__":
    # CLI Arguments passed by n8n
    script = sys.argv
    assets = sys.argv
    assemble_video(script, assets)
```

Why this approach?

- **Modularity:** By extracting logic rather than cloning the whole repo, we avoid the "bloat" of web UIs (Streamlit/Gradio) that we don't need (since n8n is our UI).
- **Control:** We can swap MoviePy for FFmpeg direct commands if performance becomes a bottleneck (MoviePy is slower but easier to code complex text effects with).[22]

---

8. Implementation Guide: The Step-by-Step Build

8.1 Step 1: Environment & Signal Detection

1. **Docker Setup:** Deploy the containers defined in Section 4.1. Ensure they share a volume.
2. **Trend Script:** Create get_trends.py.
    - *Library:* pytrends.
    - *Action:* Scrapes Google Trends.
    - *Output:* Returns the top "Rising" query in the "Science" or "Business" category.

8.2 Step 2: The n8n Workflow Construction

1. **Trigger:** Cron (Daily).
2. **Execute Command:** python3 get_trends.py.
3. **LLM Chain:**
    - **Writer:** Generates script based on Trend.
    - **Critic:** Scores script.

    - **Logic:** Loop until Score > 8.
4. **Asset Generation:**
    - **SplitInBatches:** Split script into sentences.
    - **LLM (Prompter):** Convert sentence to Visual Prompt.
    - **HTTP Request:** Send Prompt to ComfyUI API. Save outputs to /local_files/run_id/.
5. **Assembly:**
    - **Execute Command:** python3 assemble.py "{{script}}" "/local_files/run_id/".

8.3 Step 3: Optimization & "Production Grade"

To move from "Prototype" to "Production":

- **Voice Variation:** Randomize the Edge-TTS voice (en-US-ChristopherNeural vs en-US-EricNeural) to avoid channel monotony.
- **Visual Style:** Use ComfyUI's IPAdapter to enforce a consistent style across all generated clips (e.g., "Neo-Noir Comic Book Style"). This creates a brand identity without a face.[24]
- **Metadata:** Use Ollama to generate the Title, Description, and Tags based on the final script.

---

9. Conclusion: The Autonomous Future

This report outlines a system that transcends simple automation. By integrating a **Cognitive Loop (Critic-Refiner)**, we solve the quality problem that plagues algorithmic content. By leveraging **Local AI (Ollama/ComfyUI/Edge-TTS)**, we solve the cost and privacy problem. By strategically forking logic from **MoneyPrinterTurbo** and **ShortGPT**, we solve the engineering complexity problem.

The result is a self-contained, zero-cost media studio. It listens to the market, ideates concepts, verifies its own quality, synthesizes assets, and assembles the final product—all without human intervention. In the saturated market of 2026, this architectural rigor is the only path to building a sustainable, high-RPM faceless channel. The machine does not just print videos; it prints *verified* attention.

Works cited

1. I analyzed 50000 faceless YouTube channels. Here's what actually works in 2025. - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/PartneredYoutube/comments/1ooz675/i_analyzed_50000_faceless_youtube_channels_heres/
2. Top 15 Faceless Youtube Channel Ideas for 2026 - Fliki, accessed on January 4, 2026, https://fliki.ai/blog/faceless-youtube-channel-ideas
3. Top 5 Faceless YouTube Niches That Actually Make Money (2025), accessed on January 4, 2026, https://www.youtube.com/watch?v=cHS118HaVcQ
4. Top 10 Most Profitable YouTube Niches in 2026: How to Find Your Winning

Channel Idea, accessed on January 4, 2026, https://outlierkit.com/blog/most-profitable-youtube-niches
5. 30+ YouTube Channel Ideas for 2025 (Faceless & Beginner-Friendly) - Animoto, accessed on January 4, 2026, https://animoto.com/blog/video-ideas/best-youtube-channel-ideas-faceless
6. Low Competition Faceless YouTube Niches That Still Work in 2026 - ShortVids, accessed on January 4, 2026, https://shortvids.co/low-competition-faceless-youtube-niches/
7. 13 ChatGPT Prompts for YouTube Shorts [UPDATED] - LearnPrompt.org, accessed on January 4, 2026, https://learnprompt.org/chatgpt-prompts-for-youtube-shorts/
8. n8n Tutorial | Free Self-Hosted Automation for Developers, accessed on January 4, 2026, https://www.youtube.com/watch?v=lcbyKGJWp-k
9. Iterative content refinement with GPT-4 multi-agent feedback system | n8n workflow template, accessed on January 4, 2026, https://n8n.io/workflows/5597-iterative-content-refinement-with-gpt-4-multi-agent-feedback-system/
10. Agentic RAG Demystified: How n8n Workflows Make AI Retrieval Smarter | by PrajnaAI, accessed on January 4, 2026, https://prajnaaiwisdom.medium.com/agentic-rag-demystified-how-n8n-workflows-make-ai-retrieval-smarter-19b88d38289b
11. ByteDance/AnimateDiff-Lightning - Hugging Face, accessed on January 4, 2026, https://huggingface.co/ByteDance/AnimateDiff-Lightning
12. comfyui-animatediff Custom Node, accessed on January 4, 2026, https://comfyai.run/custom_node/comfyui-animatediff
13. Automating Image Generation with n8n and ComfyUI - DEV Community, accessed on January 4, 2026, https://dev.to/worldlinetech/automating-image-generation-with-n8n-and-comfyui-521p
14. Generate AI media with ComfyUI: Images, video, 3D & audio bridge | n8n workflow template, accessed on January 4, 2026, https://n8n.io/workflows/4468-generate-ai-media-with-comfyui-images-video-3d-and-audio-bridge/
15. Kosinkadink/ComfyUI-AnimateDiff-Evolved - GitHub, accessed on January 4, 2026, https://github.com/Kosinkadink/ComfyUI-AnimateDiff-Evolved
16. jacky-xbb/faceless-video-api: A FastAPI-based service that powers automated video content creation through AI. - GitHub, accessed on January 4, 2026, https://github.com/SmartClipAI/faceless-video-api
17. The Dispatch Report: GitHub Repo Analysis: harry0703/MoneyPrinterTurbo, accessed on January 4, 2026, https://thedispatch.ai/reports/5857/
18. MoneyPrinterTurbo/app/services/task.py at main - GitHub, accessed on January 4, 2026, https://github.com/harry0703/MoneyPrinterTurbo/blob/main/app/services/task.py
19. GitHub - RayVentura/ShortGPT: ShortGPT - An experimental AI framework for automated short/video content creation. Enables creators to rapidly produce,

manage, and deliver content using AI and automation. - BestofAI, accessed on January 4, 2026, https://bestofai.com/article/github-rayventurashortgpt-shortgpt-an-experimental-ai-framework-for-automated-shortvideo-content-creation-enables-creators-to-rapidly-produce-manage-and-deliver-content-using-ai-and-automation
20. ShortGPT Hello World Example, accessed on January 4, 2026, https://docs.shortgpt.ai/docs/getting-started
21. Best Open Source Video Editor SDKs: 2025 Roundup | IMG.LY Blog, accessed on January 4, 2026, https://img.ly/blog/best-open-source-video-editor-sdks-2025-roundup/
22. Is using just ffmpeg be faster than moviepy - Reddit, accessed on January 4, 2026, https://www.reddit.com/r/moviepy/comments/t1sm6k/is_using_just_ffmpeg_be_faster_than_moviepy/
23. ComfyUI AnimateDiff, ControlNet and IP-Adapter Workflow | Video2Video - RunComfy, accessed on January 4, 2026, https://www.runcomfy.com/comfyui-workflows/comfyui-animatediff-controlnet-and-ipadapter-workflow-video2video