Whop.com Video Clipping n8n Workflow

This workflow automates the process of clipping specific segments from videos hosted on Whop.com. Due to the nature of video processing, this workflow will outline the necessary steps and suggest tools, as direct n8n nodes for complex video manipulation are not standard.

Workflow Overview

1. **Trigger:** Detect a new video or a request to clip a video from Whop.com.
2. **Retrieve Video:** Download the video file from Whop.com using its API.
3. **Define Clip Segments:** Specify the start and end times for the desired clip.
4. **Process Video:** Use a video processing tool to extract the specified segment.
5. **Upload Clipped Video:** Upload the new clipped video to a desired destination (e.g., back to Whop.com, cloud storage, social media).

Detailed Steps and Suggested Tools

Phase 1: Trigger

- **Option A: Webhook Trigger (Preferred if Whop.com supports it)**
  - **n8n Node:** `Webhook`
  - **Description:** Configure a webhook in Whop.com (if available) to send a notification to n8n when a new video is uploaded or a clipping request is made. The webhook payload should ideally include the video ID or URL.
- **Option B: Polling Trigger (If webhooks are not available)**
  - **n8n Node:** `Cron` or `HTTP Request`
  - **Description:** Use a `Cron` node to periodically trigger the workflow. Within the workflow, use an `HTTP Request` node to query the Whop.com API for new videos or clipping requests. This will require storing the last checked video ID/timestamp to avoid reprocessing.

Phase 2: Retrieve Video

- **n8n Node:** `HTTP Request`

- **Description:** Use the Whop.com API (specifically, endpoints related to video content or assets) to retrieve the direct URL of the video file. You will need to handle authentication (API Key or OAuth) as per Whop.com's developer documentation. Once the URL is obtained, another `HTTP Request` node can be used to download the video file to a temporary location within the n8n environment or a connected storage service.

Phase 3: Define Clip Segments

- **Option A: Manual Input (For specific, one-off clips)**
  - **n8n Node:** `Manual Trigger` or `Webhook` with input fields
  - **Description:** The workflow can be manually triggered, or a webhook can accept parameters for `start_time` and `end_time` (e.g., in seconds or `HH:MM:SS` format). This is suitable for ad-hoc clipping.
- **Option B: Automated/AI-driven (More advanced)**
  - **n8n Node:** `Code` (Python/JavaScript) or integration with AI services
  - **Description:** This is a complex step. It would involve sending the video (or its transcript/metadata) to an AI service (e.g., for highlight detection, speech-to-text analysis to find keywords, or scene change detection). The AI service would then return the `start_time` and `end_time` for the desired clips. This would likely require custom code within n8n to interact with such services.

Phase 4: Process Video (Clipping)

- **Tool:** FFmpeg (via `Execute Command` node or external service)
- **Description:** FFmpeg is a powerful open-source command-line tool for video and audio processing. Since n8n does not have a native video clipping node, you would need to:

1. **Install FFmpeg:** Ensure FFmpeg is available in the environment where n8n is running (or use a Docker image with FFmpeg pre-installed).
2. **n8n Node:** `Execute Command`
3. **Command:** Use the `Execute Command` node to run an FFmpeg command. A basic clipping command looks like this:
   `bash`
   `ffmpeg -i input.mp4 -ss [start_time] -to [end_time] -c copy output.mp4`
   - `input.mp4`: Path to the downloaded video file.

        - `[start_time]`: Start time of the clip (e.g., `00:01:30` or `90`).
        - `[end_time]`: End time of the clip (e.g., `00:02:00` or `120`).
        - `-c copy`: Copies the video and audio streams without re-encoding, which is very fast but requires the start/end times to be on keyframes. For more precise cuts, re-encoding might be necessary (remove `-c copy`).
        - `output.mp4`: Desired name for the clipped video file.

    4. **Alternative: Cloud Video Processing API:** If direct FFmpeg execution is not feasible or scalable, integrate with a cloud video processing service (e.g., AWS Elemental MediaConvert, Google Cloud Video Intelligence API, Cloudinary, Mux) via their APIs using the `HTTP Request` node. These services typically handle the heavy lifting of video processing.

Phase 5: Upload Clipped Video

- **n8n Node:** `HTTP Request` or specific integration nodes (e.g., `Google Drive`, `Dropbox`, `S3`, `YouTube`)
- **Description:** After the video is clipped, upload the `output.mp4` file to its final destination. This could be:
    - **Whop.com:** If Whop.com provides an API for uploading user-generated content or assets.
    - **Cloud Storage:** Upload to a cloud storage bucket (e.g., Amazon S3, Google Cloud Storage) for hosting.
    - **Social Media:** Upload to platforms like YouTube, Vimeo, or directly to social media if n8n has dedicated nodes or if their APIs are accessible via `HTTP Request`.

Considerations

- **File Storage:** Temporary storage for downloaded and clipped videos within the n8n environment. For large files, consider external storage solutions.
- **Error Handling:** Implement robust error handling for API calls, video processing failures, and upload issues.
- **Scalability:** For high-volume clipping, consider running n8n in a scalable environment or offloading video processing to dedicated cloud services.
- **Authentication:** Securely manage API keys and credentials for Whop.com and any other integrated services.

This outline provides a conceptual framework. The exact implementation will depend on the specific capabilities of the Whop.com API and the chosen video processing method.