Building a Cloud-Based Automated Daily News Digest

Using Claude AI, Make (Integromat), and Gmail

A Technical Portfolio Paper

Michael Rahl

April 2026

1. Abstract

This paper documents the design and implementation of a fully cloud-based automation system that delivers two personalized daily news digest emails — one covering cryptocurrency investment and one covering artificial intelligence — directly to a Gmail inbox every morning at 7:00 AM.

The system uses the Claude Opus 4 language model via the Anthropic API to generate formatted HTML email content, Make (formerly Integromat) as the cloud automation and scheduling layer, and Gmail as the delivery mechanism. The result is a zero-maintenance, always-on personal news briefing that requires no manual intervention.

2. Objective

The objective of this project was to build a system that automatically delivers two curated news digest emails each morning — one for crypto investment news and one for AI news — with no manual effort required on any given day.

The system was designed to meet the following requirements:

- Fully automated — no manual triggering required each day
- Cloud-based — no dependency on a local computer being powered on
- Delivered directly to a Gmail inbox
- Rich HTML formatting with clickable article title hyperlinks
- Two separate emails: one for crypto investment news, one for AI news
- Each email to contain at least 7 curated articles with descriptions

3. System Architecture

The system is built as a single Make scenario containing four sequential modules that execute in order each morning:

1. **HTTP Module 1 (Crypto Request):** Sends a prompt to the Claude API requesting a crypto investment digest in HTML format
2. **Gmail Module 1 (Crypto Email):** Takes the HTML response from Module 1 and sends it as an email to Gmail
3. **HTTP Module 2 (AI Request):** Sends a prompt to the Claude API requesting an AI news digest in HTML format
4. **Gmail Module 2 (AI Email):** Takes the HTML response from Module 3 and sends it as an email to Gmail

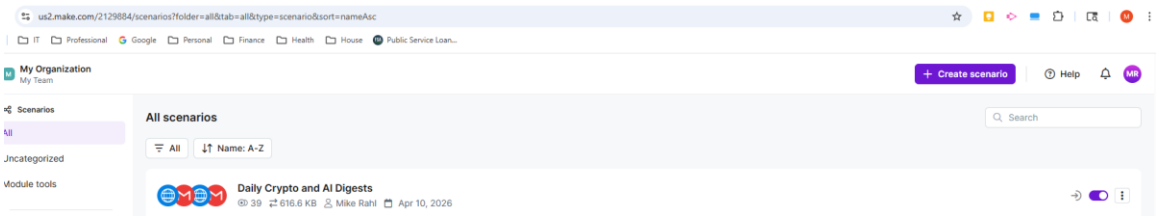

Figure 1: Make scenarios dashboard showing the Daily Crypto and AI Digests scenario.

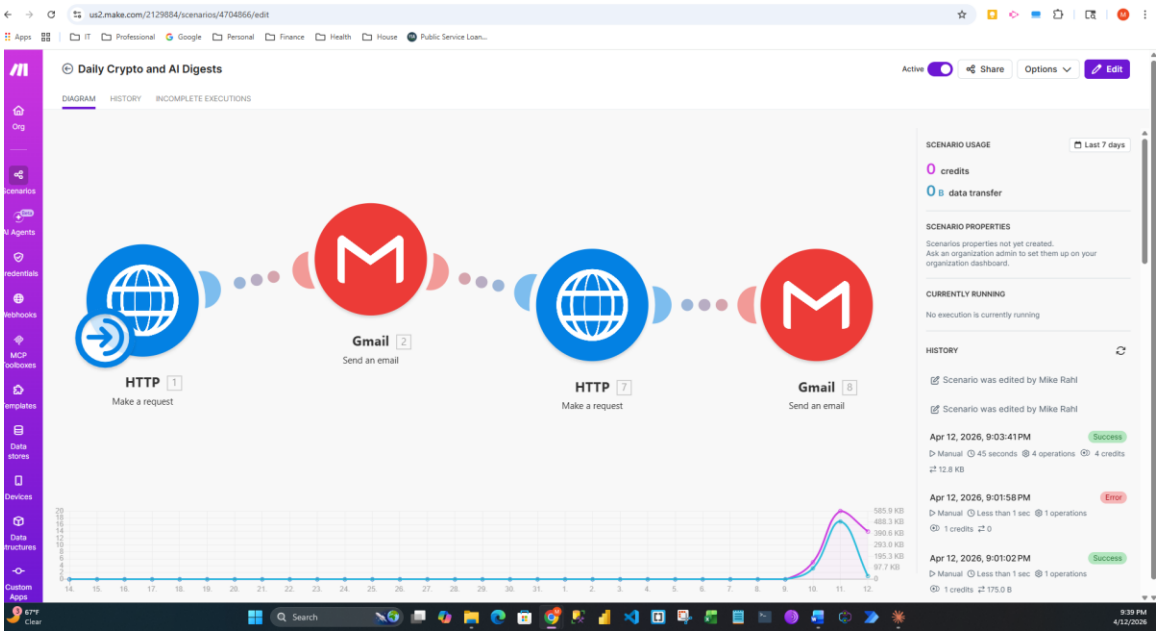

Figure 2: Full Make scenario canvas — HTTP → Gmail → HTTP → Gmail modules connected in sequence.

The scenario is triggered by Make's built-in cloud scheduler, set to run daily at 7:00 AM Eastern Time. Because Make runs entirely in the cloud, the automation fires at the scheduled time regardless of whether any local device is powered on or connected.

Technology stack:

- Anthropic Claude Opus 4 (claude-opus-4-6) — content generation via REST API
- Make (Integromat) — cloud automation, scheduling, and module orchestration
- Gmail — email delivery via OAuth 2.0 authenticated connection
- Anthropic Messages API: https://api.anthropic.com/v1/messages

4. Implementation

4.1 Gmail Connection Setup

The first step was establishing a Gmail OAuth 2.0 connection within Make to authorize sending emails on behalf of michaelrahl@gmail.com. The connection was authorized through Google's standard OAuth consent flow and saved as a reusable connection within the Make workspace.

This connection is used by both Gmail modules in the scenario and must be reauthorized periodically as required by Google's OAuth policies.

4.2 HTTP Module Configuration (Claude API)

Each HTTP module is configured to make a POST request to the Anthropic Messages API. The settings are identical for both modules except for the prompt content.

URL:

```
https://api.anthropic.com/v1/messages
```

Method:

```
POST
```

Headers:

```
x-api-key: [Anthropic API Key]
anthropic-version: 2023-06-01
content-type: application/json
```

Parse response: Enabled

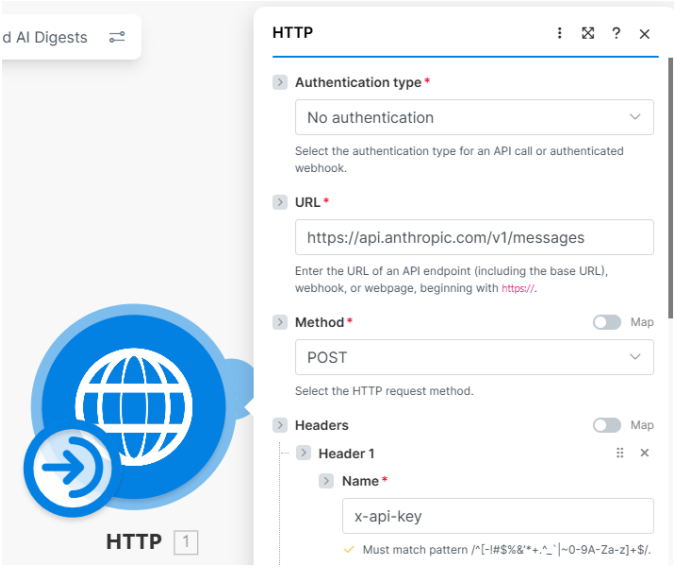

Figure 3: HTTP module — authentication type, URL, and method configuration.

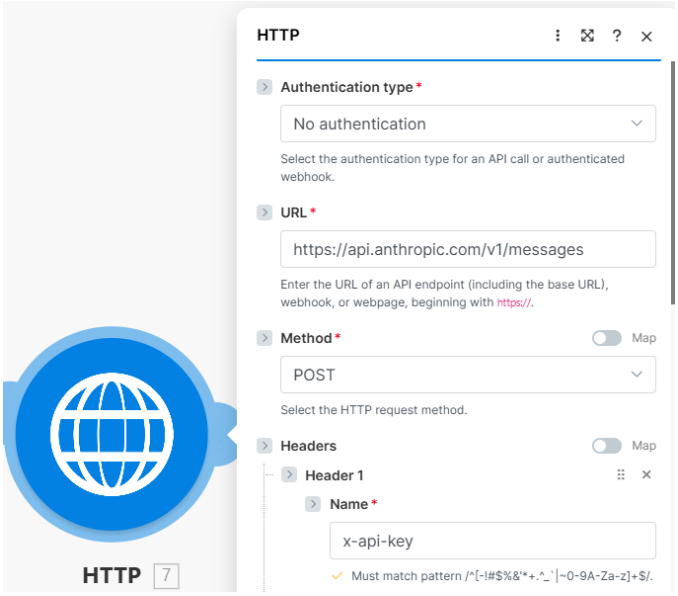

Figure 4: HTTP module — headers and JSON body configuration.

Request body — Crypto module:

```
{
  "model": "claude-opus-4-6",
  "max_tokens": 4096,
  "messages": [{
    "role": "user",
    "content": "Write a daily crypto investment digest email for today.
    Return ONLY raw HTML with no markdown, no code blocks, no backticks.
    Start directly with <html> and end with </html>.
    Include a 2-3 sentence market summary intro, then a bulleted list
    of 7 articles. For each article make the title a clickable hyperlink
    using this format:
    <li><a href='ARTICLE_URL'>Article Title</a> - one-line description.</li>"
  }]
}
```

Request body — AI module:

```
{
  "model": "claude-opus-4-6",
  "max_tokens": 4096,
  "messages": [{
    "role": "user",
    "content": "Write a daily AI news digest email for today.
    Return ONLY raw HTML with no markdown, no code blocks, no backticks.
    Start directly with <html> and end with </html>.
    Include a 2-3 sentence summary of the most significant AI developments,
    then a bulleted list of 7 articles. For each article make the title a
    clickable hyperlink using this format:
    <li><a href='ARTICLE_URL'>Article Title</a> - one-line description.</li>"
  }]
}
```

The prompt instructs Claude to return raw HTML only — no markdown, no code fences, no explanatory text. Specifying the exact HTML element structure for each list item ensures the output is directly usable as an email body without any post-processing.

4.3 Gmail Module Configuration

Each Gmail module is configured to send an email using the HTML body returned by the preceding HTTP module. The Content field uses a Make variable formula that references the HTTP module's response.

Crypto Gmail module settings:

- **To:** michaelrahl@gmail.com
- **Subject:** 🪙 Daily Crypto Digest — {{formatDate(now; "MMMM D, YYYY")}}
- **Body type:** Raw HTML
- **Content:** {{1.data.content[1].text}}

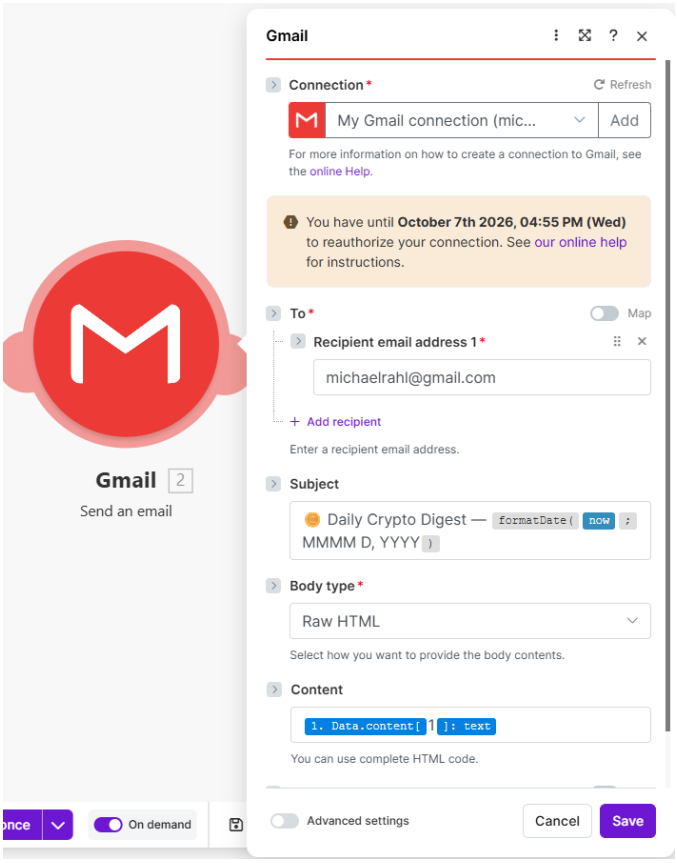

Figure 5: Gmail module configuration for the Crypto Digest email.

AI Gmail module settings:

- **To:** michaelrahl@gmail.com
- **Subject:** 🤖 Daily AI Digest — {{formatDate(now; "MMMM D, YYYY")}}
- **Body type:** Raw HTML

- **Content:** {{7.data.content[1].text}}

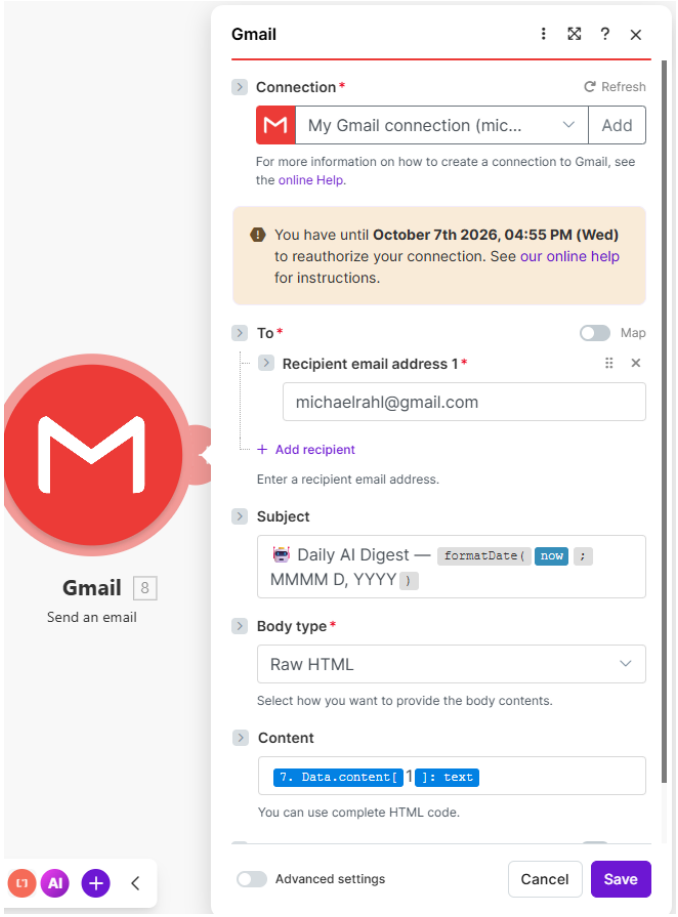

Figure 6: Gmail module configuration for the AI Digest email.

The variable {{1.data.content[1].text}} references the first item of the content array returned by the Claude API (Make uses 1-based array indexing). The formatDate function dynamically inserts today's date into the subject line on each run.

4.4 Scheduling

The scenario is activated and scheduled using Make's built-in scheduler, set to run daily at 7:00 AM Eastern Time. The toggle at the top of the scenario view activates the scenario and the schedule is configured via the scheduling controls at the bottom of the canvas.

Because the scheduler runs on Make's servers, the automation is entirely independent of any local device. The emails will be delivered at 7:00 AM every day as long as the scenario remains active and the Gmail connection is authorized.

5. Results

The completed system successfully delivers two HTML-formatted digest emails to michaelrahl@gmail.com every morning at 7:00 AM Eastern Time with no manual intervention required.

Each email contains:

- A 2-3 sentence summary of the day's market or technology landscape
- 7 curated articles with titles hyperlinked directly to the source
- A one-line description for each article
- A dynamic date in the subject line (e.g., "Daily Crypto Digest — April 12, 2026")

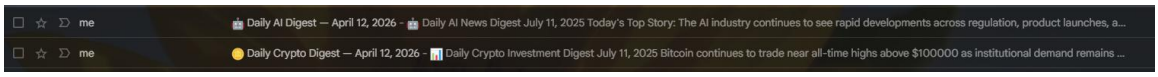

Figure 7: Gmail inbox showing both the Crypto and AI digest emails delivered automatically.

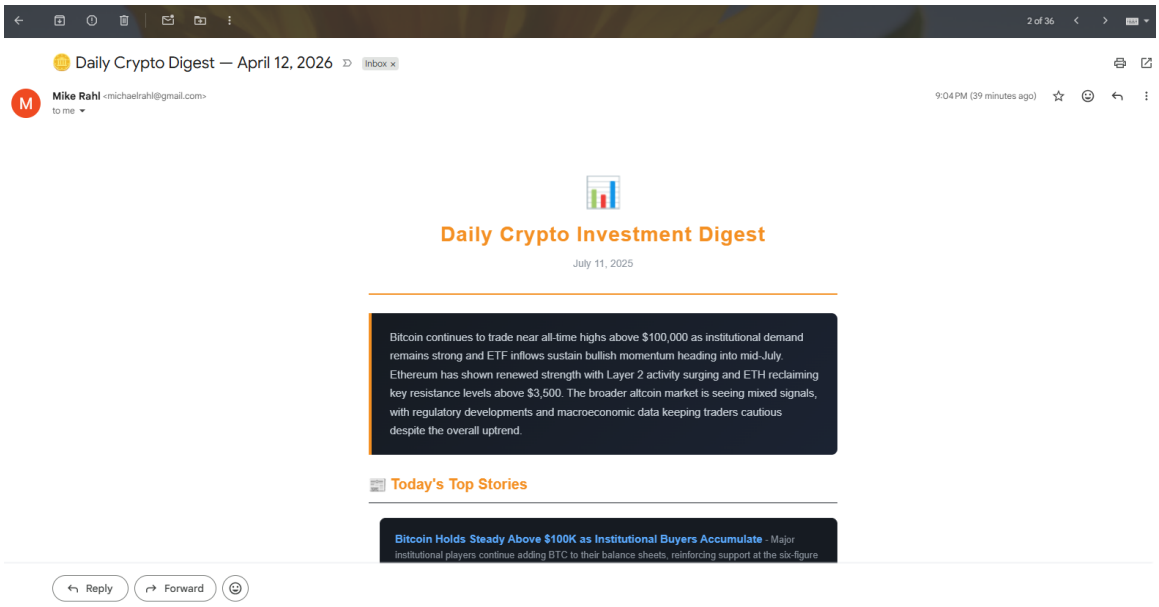

Figure 8: Daily Crypto Investment Digest email — formatted HTML with market summary and hyperlinked articles.

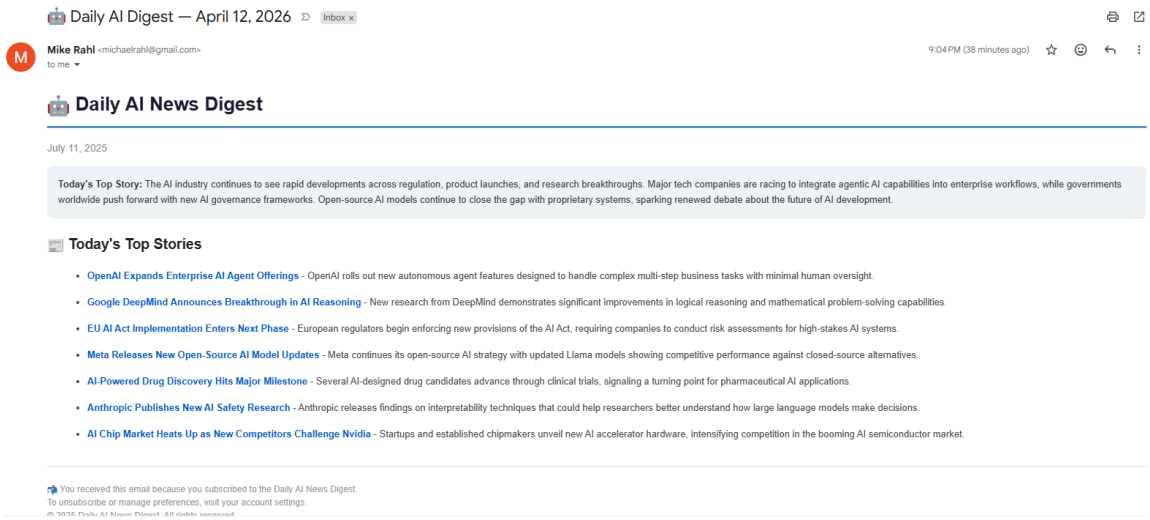

Figure 9: Daily AI News Digest email — formatted HTML with AI developments summary and hyperlinked articles.

The scenario consumes approximately 4 Make operations per day (2 HTTP calls + 2 Gmail sends), totaling roughly 120 operations per month — well within Make's free tier allowance of 1,000 operations per month.

6. Conclusion

This project demonstrates a practical, production-ready application of large language model APIs as dynamic content generators within a cloud automation pipeline. By combining Claude's natural language capabilities with Make's scheduling infrastructure and Gmail's delivery system, a fully automated personal news briefing was built that requires zero ongoing maintenance.

The architecture is intentionally simple and extensible. Additional digest topics — such as finance, sports, or technology news — can be added to the same Make scenario by appending new HTTP and Gmail module pairs without modifying the existing flow.

Key design principles applied in this project:

- Precise prompt engineering ensures Claude returns output in the exact format the downstream system expects, eliminating the need for any post-processing or transformation steps.
- Cloud-native scheduling decouples the automation from any local device, making the system reliable and always-on.
- Single quotes in HTML attribute examples within JSON request bodies avoid character escaping conflicts, keeping the payload valid across all environments.
- Dynamic date injection via formatDate keeps email subjects current and informative without any manual updates.

Appendix A: Final Prompt Templates

Crypto Investment Digest Prompt

```
Write a daily crypto investment digest email for today. Return ONLY raw HTML with no markdown, no code blocks, no backticks. Start directly with <html> and end with </html>. Include a 2-3 sentence market summary intro, then a bulleted list of 7 articles. For each article make the title a clickable hyperlink using this format: <li><a href='ARTICLE_URL'>Article Title</a> - one-line description.</li>
```

AI News Digest Prompt

```
Write a daily AI news digest email for today. Return ONLY raw HTML with no markdown, no code blocks, no backticks. Start directly with <html> and end with </html>. Include a 2-3 sentence summary of the most significant AI developments, then a bulleted list of 7 articles. For each article make the title a clickable hyperlink using this format: <li><a href='ARTICLE_URL'>Article Title</a> - one-line description.</li>
```

Appendix B: Make Variable Reference

The following Make variable formulas are used in the scenario:

- {{1.data.content[1].text}} — HTML body from HTTP Module 1 (Crypto)
- {{7.data.content[1].text}} — HTML body from HTTP Module 7 (AI)
- {{formatDate(now; "MMMM D, YYYY")}} — Current date, e.g. "April 12, 2026"