HOW-TO: Building a Daily Summarizer with Claude Code

A guide to creating automated daily summaries from project management tools (Asana, ClickUp, etc.) using Claude Code's conversational workflow.

Overview

This document describes how to build a daily summarizer that:

1. Fetches data from Asana (task movements, assignments, comments)
2. Generates a structured report
3. Optionally cleans up duplicates using Claude headless mode
4. Posts to ClickUp chat channels
5. Runs automatically via cron

Time to build: ~30 minutes of conversation with Claude Code

Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│                      Daily Summarizer Flow                        │
├─────────────────────────────────────────────────────────────────┤
│                                                                  │
│   Asana API ───────┐                                             │
│   (tasks, stories)│                                              │
│                    ▼                                             │
│             ┌─────────────┐                                      │
│             │ Bash Script │                                      │
│             │ (curl + jq) │                                      │
│             └──────┬──────┘                                      │
│                    │                                             │
│                    ▼                                             │
│             ┌─────────────┐     ┌─────────────┐                  │
│             │ Raw Report  │────▶│ Claude -p   │ (optional)       │
│             │ (stdout)    │     │ (dedupe)    │                  │
│             └──────┬──────┘     └──────┬──────┘                  │
│                    │                   │                         │
│                    ▼                   ▼                         │
│             ┌─────────────┐     ┌─────────────┐                  │
│             │ .md file   │    │ ClickUp API │ (optional)        │
│             │ (saved)     │     │ (post msg)  │                  │
│             └─────────────┘     └─────────────┘                  │
│                                                                  │
└─────────────────────────────────────────────────────────────────┘
```

Step-by-Step Prompting Guide

Phase 1: Generate API Skills (Optional but Recommended)

First, create API skills so Claude understands the APIs you're working with.

Prompt:

```
Generate Asana skill from https://developers.asana.com/reference/rest-api-reference.
Explore all other additional pages in the webpage instead of this URL only.
```

What happens:

- Claude uses WebFetch to scrape API documentation
- Creates a SKILL.md file with endpoints, authentication, rate limits
- May create helper Python scripts

Repeat for other APIs:

```
Generate ClickUp skill from https://developer.clickup.com/reference/getaccesstoken.
Explore all other additional pages in the webpage instead of this URL only.
```

Phase 2: Initial Data Exploration

Prompt:

```
Give me a summary of the ticket movements, assignment details, comments and
updates to the tasks in [PROJECT_NAME] board. Always include the action user
for this. Use the asana skill. This is for the last 24 hours only.
```

What Claude does:

1. Loads the Asana skill
2. Discovers workspace and project IDs via API
3. Fetches tasks modified in last 24 hours
4. Retrieves "stories" (activity feed) for each task
5. Presents raw data

Key insight: Claude learns your project structure through exploration. Let it discover IDs rather than hardcoding them.

Phase 3: Iterative Refinement

This is where the magic happens. Review Claude's output and refine through conversation.

Example refinements:

```
# Request specific format
"Summarize this in a cleaner format"

# Identify what matters
"To do column is not so important in this case"

# Add metrics
"Add workload distribution. Also, add deadline passed along with days"

# Exclude irrelevant items
"Tasks in submitted doesn't need to be in Overdue"

# Request specific layout
"Keep it simple with this format: [paste example]"
"in this format instead of creating tables"
```

Pattern: Show Claude an example of what you want, and it will match that format.

Phase 4: Script Generation

Once the format is finalized:

Prompt:

```
Create a script that generates this daily summary automatically
```

Claude generates:

- A bash script using curl + jq
- Proper error handling
- Date calculations

- Temp file management

Phase 5: Add Post-Processing

Prompt:

```
Add post processing to this using claude headless and then save to
asana_daily_summary/{date}.md file
```

What this adds:

- Calls `claude -p "prompt" --output-format text`
- Deduplicates entries (same task appearing multiple times)
- Cleans up formatting

Important: Keep the Claude prompt simple and specific:

```
"Clean up this report by deduplicating entries. Keep the EXACT same format.
Do NOT convert to tables. Do NOT add executive summary."
```

Phase 6: Add Integrations

Prompt:

```
Add another post processing to this script to post this to the ClickUp
#Scraping channel. Use ClickUp skill for this.
```

What this adds:

- ClickUp API call to post messages
- Channel ID discovery
- Success/failure handling

Phase 7: Add Options

Prompt:

```
Keep it simple with this format. Add options to toggle post-processing
and clickup posts
```

Result: Script now has flags:

```
./script.sh         # Raw only
./script.sh -p      # + Claude post-processing
./script.sh -c      # + ClickUp posting
./script.sh -a      # All options
```

Phase 8: Automation

Prompt:

```
Set up cron to run daily at 8am. This includes posting to clickup with the -a args.
```

What Claude creates:

1. A cron wrapper script (handles PATH, logging)
2. Adds entry to crontab
3. Sets up log file location

Key Prompting Patterns

1. Show, Don't Tell

Instead of: "Make the output cleaner"

Do this: Paste an example of the exact format you want

```
Keep it simple with this format:
📊 Title
✅ COMPLETED
   • Task → State (User)
...
```

2. Iterative Narrowing

Start broad, then narrow down:

```
1. "Give me a summary of all activity"
2. "Focus on these sections: ..."
3. "Remove duplicates"
4. "Add these specific metrics"
```

3. Explicit Exclusions

Be specific about what NOT to do:

```
"Do NOT convert to tables"
"Do NOT add executive summary"
"Tasks in submitted doesn't need to be in Overdue"
```

4. Skill Loading

Prefix API tasks with skill invocation:

```
"Use the asana skill to..."
"Use the clickup skill for this"
```

5. Environment Discovery

Let Claude discover IDs rather than hardcoding:

```
# Good - Claude finds the ID
"Get summary of Ideon Scraping Internal board"

# Less flexible - hardcoded
"Use project ID 1208639428824137"
```

Forking for Different Use Cases

Different Project Management Tool

1. Generate a skill for your tool (Jira, Linear, Monday, etc.)
2. Start with: "Give me a summary of [project] activity for last 24 hours"
3. Refine format through conversation
4. Request script generation

Different Output Destination

Replace ClickUp posting with:

- Slack: Use Slack API skill
- Email: Use SMTP or email API
- Notion: Use Notion API
- Discord: Use Discord webhooks

Different Metrics

Modify the prompts to focus on:

- Sprint velocity
- Bug counts
- Review times
- Deployment frequency

File Structure

```
your-project/
├── asana_daily_summary.sh          # Main script
├── asana_daily_summary_cron.sh     # Cron wrapper
├── asana_daily_summary/            # Output directory
│   ├── 2026-01-13.md              # Daily reports
│   └── cron.log                   # Execution logs
├── .claude/
│   └── skills/
│       ├── asana/SKILL.md         # Asana API reference
│       └── clickup/SKILL.md       # ClickUp API reference
└── README.md
```

Environment Setup

Required Environment Variables

```
# In ~/.claude/.env or project .env

# Asana (https://app.asana.com/0/developer-console)
ASANA_ACCESS_TOKEN=your_personal_access_token

# ClickUp (Settings > Apps > API Token)
CLICKUP_API_TOKEN=pk_your_token

# Add others as needed
SLACK_BOT_TOKEN=xoxb-...
```

Testing Tokens

```
# Test Asana
curl -s "https://app.asana.com/api/1.0/users/me" \
  -H "Authorization: Bearer $ASANA_ACCESS_TOKEN" | jq

# Test ClickUp
curl -s "https://api.clickup.com/api/v2/team" \
  -H "Authorization: $CLICKUP_API_TOKEN" | jq
```

Troubleshooting

Claude headless mode fails

```
# Ensure claude is in PATH
which claude

# Test directly
echo "hello" | claude -p "respond with hi"
```

Duplicate entries in output

Add stronger deduplication prompt:

```
"If a task appears multiple times, keep ONLY the final state reached"
```

Cron not running

```bash
# Check crontab
crontab -l

# Test wrapper manually
./asana_daily_summary_cron.sh

# Check logs
tail -100 asana_daily_summary/cron.log
```

API rate limits

- Asana: 1500 requests/minute
- ClickUp: 100 requests/minute (free), 10000 (paid)

Add delays between API calls if hitting limits.

Example Conversation Flow

Here's the actual conversation sequence that built this summarizer:

```
User: Give me a summary of ticket movements in Ideon Scraping Internal board.
      Use the asana skill. Last 24 hours only.

Claude: [Fetches data, presents raw output]

User: Summarize this

Claude: [Creates summarized format with sections]

User: Look at the board and suggest what fields would be better for a daily summary

Claude: [Analyzes board structure, suggests sections]

User: To do column is not so important. These are also important [pastes example]

Claude: [Refines to match user's preferred format]

User: Add workload distribution. Also, add deadline passed along with days

Claude: [Adds overdue tracking and workload metrics]

User: Tasks in submitted doesn't need to be in Overdue

Claude: [Excludes submitted tasks from overdue tracking]

User: yes [to creating script]

Claude: [Generates bash script]

User: Add post processing using claude headless

Claude: [Adds claude -p call for deduplication]

User: Add posting to ClickUp #Scraping channel

Claude: [Adds ClickUp API integration]

User: Keep it simple with this format. Add options to toggle features.
```

```
Claude: [Adds --process, --clickup, --all flags]

User: Set up cron to run daily at 8am

Claude: [Creates cron wrapper, adds to crontab]

User: git commit

Claude: [Commits all files with descriptive message]
```

Total: ~20 conversational turns to build a production-ready daily summarizer

---

Summary

Building a daily summarizer with Claude Code follows this pattern:

1. **Generate skills** for APIs you'll use
2. **Explore data** with broad queries
3. **Refine format** iteratively (show examples of what you want)
4. **Generate script** once format is finalized
5. **Add integrations** (post-processing, notifications)
6. **Add options** for flexibility
7. **Automate** with cron

The key is treating it as a conversation where you progressively refine the output until it matches exactly what you need.