Overview

This guide walks through building a Bun-based Gmail automation to mark mailing list emails as read once they are older than 7 days. It uses the Gmail API, compiles into a single binary, and runs via cron.

Project Structure

gmail-cleaner/
- index.ts
- credentials.json
- token.json

Install Dependencies

Run:

bun add googleapis

Full Script (index.ts)

```
import { google } from "googleapis";
import fs from "fs";
import http from "http";
import url from "url";

const SCOPES = ["https://www.googleapis.com/auth/gmail.modify"];
const TOKEN_PATH = "token.json";
const CREDENTIALS_PATH = "credentials.json";

function loadCredentials() {
const content = fs.readFileSync(CREDENTIALS_PATH, "utf-8");
const { client_secret, client_id, redirect_uris } = JSON.parse(content).installed;
return new google.auth.OAuth2(client_id, client_secret, redirect_uris[0]);
}

async function authorize() {
const auth = loadCredentials();

if (fs.existsSync(TOKEN_PATH)) {
const token = JSON.parse(fs.readFileSync(TOKEN_PATH, "utf-8"));
auth.setCredentials(token);
return auth;
}

const authUrl = auth.generateAuthUrl({
access_type: "offline",
```

```
scope: SCOPES,
});

console.log("Visit this URL:");
console.log(authUrl);

return new Promise((resolve) => {
const server = http.createServer(async (req, res) => {
const qs = new url.URL(req.url!, "http://localhost:3000").searchParams;
const code = qs.get("code");

if (code) {
const { tokens } = await auth.getToken(code);
auth.setCredentials(tokens);

fs.writeFileSync(TOKEN_PATH, JSON.stringify(tokens));
res.end("Auth successful. You can close this window.");
server.close();

resolve(auth);
}
});

server.listen(3000);
});
}

async function markOldEmails(auth: any) {
const gmail = google.gmail({ version: "v1", auth });

const query = "label:mailing-list older_than:7d is:unread -is:starred -is:important";

const res = await gmail.users.threads.list({
userId: "me",
q: query,
});

const threads = res.data.threads || [];

if (threads.length === 0) {
console.log("No threads to process.");
return;
}

await gmail.users.threads.batchModify({
userId: "me",
requestBody: {
ids: threads.map(t => t.id!),
removeLabelIds: ["UNREAD"],
},
});

console.log(`Marked ${threads.length} threads as read.`);
}

(async () => {
```

```
const auth = await authorize();
await markOldEmails(auth);
})();
```

Google Cloud Setup

1. Create project in Google Cloud Console
2. Enable Gmail API
3. Create OAuth credentials (Desktop app)
4. Download credentials.json

First Run

Run:

```
bun run index.ts
```

Complete OAuth flow → token.json saved.

Compile to Binary

Run:

```
bun build index.ts --compile --outfile gmail-cleaner
```

Cron Job

```
0 2 * * * /opt/gmail-cleaner/gmail-cleaner >> /var/log/gmail-cleaner.log 2>&1
```

Recommended Gmail Filter

Create a Gmail filter:
list:newsletter@example.com → apply label: mailing-list

Optional Enhancements

- Dry run mode
- Multiple rules
- Slack notifications
- Containerization
- Metrics + logging