




Makefile API testing

Perfect — since the API isn't live yet, your testing setup should focus on:

-  **Prism mock server** (OpenAPI contract-based local simulation)
-  **Postman mock server** (Postman's hosted mock environment)
-  **Newman test runs** against both mock environments

You can ignore `test-live` for now — or leave it as a placeholder for later.

Adjusted Makefile (no live API yet)

Here's the simplified version without `test-live`, and with an optional `test-all` to run both Prism and Postman tests:

```
makefile

# === CONFIGURATION ===

SPEC           := openapi/c2m_openapi_spec_final.yaml
COLL_RAW       := postman/Click2Mail.collection.json
COLL_FINAL     := postman/Click2Mail.collection.merged.json
OVERRIDE_FILE  := postman/overrides.json
ENV_FILE       := postman/mock-env.json
MOCK_URL_FILE  := postman/mock-url.txt
PRISM_PORT     := 4010
REPORT_HTML    := newman-report.html

GENERATOR      := npx @apideck/postman-generator
MERGER         := node scripts/merge-postman.js
PRISM          := npx @stoplight/prism-cli
NEWMAN         := npx newman

# === TARGETS ===

.PHONY: all generate merge prism prism-stop test-prism test-postman test test-all

all: generate merge test

## Generate Postman collection from OpenAPI spec
generate: $(COLL_RAW)

$(COLL_RAW): $(SPEC)
    mkdir -p $(dir $@)
    $(GENERATOR) -s $(SPEC) -o $@ -p -O folderStrategy=Tags
    @echo "✅ Generated: $@"

## Merge collection with override (optional)
merge: $(COLL_FINAL)

$(COLL_FINAL): $(COLL_RAW) $(OVERRIDE_FILE)
    $(MERGER) -c $(COLL_RAW) -o $(OVERRIDE_FILE) -r $@
    @echo "✅ Merged with overrides: $@"

## Start Prism mock server
prism:
    @echo "🚀 Starting Prism on port $(PRISM_PORT)..."
    $(PRISM) mock $(SPEC) -p $(PRISM_PORT)

## Stop Prism mock server
prism-stop:
```

```
@echo "🔴 Killing Prism if running..."
@lsof -ti tcp:${PRISM_PORT} | xargs kill -9 || echo "No Prism running"

## Run Newman tests against Prism
test-prism:
@echo "📦 Running Newman tests against Prism..."
$(NEWMAN) run $(COLL_FINAL) \
  --env-var baseUrl=http://localhost:${PRISM_PORT} \
  --env-var token=$(shell jq -r '.variable[]|select(.key=="token").value'
$(ENV_FILE)) \
  --reporters cli,html \
  --reporter-html-export $(REPORT_HTML)

## Run Newman tests against Postman mock server
test-postman:
@echo "📦 Running Newman tests against Postman mock server..."
$(eval MOCK := $(shell cat $(MOCK_URL_FILE)))
$(NEWMAN) run $(COLL_FINAL) \
  --env-var baseUrl=$(MOCK) \
  --env-var token=$(shell jq -r '.variable[]|select(.key=="token").value'
$(ENV_FILE)) \
  --reporters cli,html \
  --reporter-html-export $(REPORT_HTML)

## Default test: use Prism
test: test-prism

## Run both Prism and Postman mock tests
test-all: test-prism test-postman
@echo "✅ All mock tests completed"
```

✅ Example Usage

```
bash

make prism          # Start Prism server
make test           # Test against Prism
make test-postman   # Test against Postman mock
make test-all       # Run both sets of tests
```

Let me know if you'd like a target that waits for Prism to be ready before running the test.



what about running newman against the actual local collection?

Great follow-up!

You **can absolutely run Newman directly against your local Postman collection** (no mock server needed), as long as the requests in that collection point to valid URLs — for example:

- Prism mock server (<http://localhost:4010>)
- Postman mock server (<https://<mock-id>.mock.pstmn.io>)
- Eventually: your real API (once it's live)

✅ Target to Run Against Local Collection

Add this target to your **Makefile**:

```
makefile
```

```
## Run Newman tests directly using local collection and env
```

```
test-local:
```

```
    @echo "📁 Running Newman tests on local collection using env values..."
    $(NEWMAN) run $(COLL_FINAL) \
        --environment $(ENV_FILE) \
        --reporters cli,html \
        --reporter-html-export $(REPORT_HTML)
```

This uses:

- `$(COLL_FINAL)` → your generated Postman collection file
- `$(ENV_FILE)` → a full Postman environment JSON (can include `baseUrl`, `token`, etc.)

✅ Sample `mock-env.json` (Postman environment format)

```
json

{
  "id": "c2m-env-id",
  "name": "C2M Local Dev",
  "values": [
    { "key": "baseUrl", "value": "http://localhost:4010", "enabled": true },
    { "key": "token", "value": "mock-token", "enabled": true }
  ],
  "_postman_variable_scope": "environment",
  "_postman_exported_at": "2025-07-16T12:00:00Z",
  "_postman_exported_using": "Postman/10.23.0"
}
```

✅ Example Command

```
bash

make test-local
```

This will:

- Use your local collection
- Substitute the environment variables (`baseUrl`, `token`)
- Run all request + test scripts in Postman
- Output a CLI and HTML report

Let me know if you'd like a target that dynamically **sets the base URL via `--env-var` instead** (useful when not using a full Postman environment JSON).



so newman cannot do anything unless it is pointing to a mock server or the collection file requests are pointing to real urls

Exactly — you've got it.

✓ How Newman Works

Newman executes requests **exactly as they appear in your collection**, meaning:

- Each request must have a valid, reachable URL — real or mock.
- It doesn't "simulate" responses — it **sends real HTTP requests**.

🚫 What Newman doesn't do:

- Newman does **not mock** requests itself.
- It won't "stub" or "simulate" responses if your server isn't running.
- If your `baseUrl` is something like `http://localhost:4010` and **nothing is listening on that port**, requests will fail.

✓ What You Can Do:

Scenario	What to Point baseUrl To
API not implemented yet	Prism mock server (<code>http://localhost:4010</code>)
Hosted mock needed	Postman mock server (<code>https://xxxx.mock.pstmn.io</code>)
Testing live service	Your actual staging/prod URL
Static example testing only	Use Prism + OpenAPI example: blocks

🔄 If You Don't Want to Spin Up a Server

If you want **mock behavior without spinning up Prism**, use:

- **Postman mock server** (easy, hosted)
- Or Prism via `make prism` for fast local mocking



Newman + Postman mocks or Prism gives you a realistic test environment even before the backend is built.

Let me know if you want help generating a minimal OpenAPI spec + example block to feed into Prism or Postman for simulated responses!



Static example testing only Use Prism + OpenAPI example: blocks

Perfect — this is a great approach when your backend isn't live yet. Let's walk through how to do **static example-based testing** using:

-  **Prism** (mock server that returns **example:** values from your OpenAPI spec)
-  **Newman** (running your Postman collection against the Prism server)

How Static Example Testing Works

Flow:

1. You define **example:** or **examples:** in your OpenAPI spec
2. Prism reads your OpenAPI file and **serves responses from those examples**
3. Newman sends real HTTP requests to Prism (e.g., **localhost:4010**)
4. Newman tests validate the responses against expectations

1. Add Examples to Your OpenAPI Spec

Here's an example for a **POST /jobs/submit/single/doc** endpoint:

```
yaml

paths:
  /jobs/submit/single/doc:
    post:
      summary: Submit single doc
      tags: [Job Submission]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Job'
            example:
              jobId: "abc123"
              printOption: "color"
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Job'
              example:
                jobId: "abc123"
                status: "submitted"
```

 Prism will now return:

- The request example as a default body when you generate Postman requests (if you're using **@apideck/postman-generator**)
- The response example (**status: submitted**) as its response payload

2. Start Prism Mock Server

```
bash

make prism
```

This runs:

```
bash

npx @stoplight/prism-cli mock openapi/c2m_openapi_spec_final.yaml -p 4010
```

✓ 3. Point Your Postman Environment at Prism

In `mock-env.json`:

```
json

{
  "values": [
    { "key": "baseUrl", "value": "http://localhost:4010", "enabled": true }
  ]
}
```

Each Postman request should use `{{baseUrl}}` in the URL.

✓ 4. Run Newman Against Prism

```
bash

make test-prism
```

Or manually:

```
bash

npx newman run postman/Click2Mail.collection.merged.json \
  --environment postman/mock-env.json \
  --reporters cli
```

You'll get **real HTTP responses, generated from static OpenAPI examples** — perfect for contract-first testing and debugging without any backend code running.

✓ Summary

Tool	Purpose
OpenAPI	Define request/response schemas + examples
Prism	Serve the example responses as a local mock server
Postman	Define and organize API requests/tests
Newman	Run the Postman tests via CLI

Let me know if you want me to scan your OpenAPI and help auto-insert good **example:** blocks!