To install all dependencies listed in a requirements.txt file, you use pip from your command line or terminal. Here’s how:

1. Make sure your requirements.txt file exists in your project directory.
2. Run the following command:

bash

CopyEdit

pip install -r requirements.txt

This will read all packages listed in requirements.txt (like requests, tweepy, atproto) and install them in your Python environment.

If you’re using a virtual environment (recommended), make sure it’s activated before running the comman

**python test\_run.py**

**Vitality Bot**

## Railway for hostin the instance

## ## 1) Project Structure

## health-longevity-bot/

## ├─ main.py # Entry point; scheduling & run loop

## ├─ bot.py # Core logic (fetch, choose, caption, image, post)

## ├─ sources.py # NewsAPI & Reddit helpers

## ├─ imaging.py # Image download & fallback image generation

## ├─ posting.py # Multi-platform posting (Twitter/X, Instagram, Facebook, Bluesky)

## ├─ requirements.txt # Python deps

## ├─ Procfile # Railway worker definition

## ├─ .env.example # Example environment variables

## └─ README.md # (optional) this doc

## ---2) Environment Variables (Railway → Variables)

Create these keys in Railway (Settings → Variables):

**Required**

* NEWSAPI\_KEY — your NewsAPI key
* REDDIT\_CLIENT\_ID — Reddit app client id
* REDDIT\_CLIENT\_SECRET — Reddit app secret
* REDDIT\_USER\_AGENT — something like longevity-curator/1.0 by <yourname>
* TWITTER\_API\_KEY — X API key
* TWITTER\_API\_SECRET — X API secret
* TWITTER\_ACCESS\_TOKEN — X access token
* TWITTER\_ACCESS\_SECRET — X access token secret

**Optional (images/time)**

* UNSPLASH\_ACCESS\_KEY — Unsplash API key (fallback image search)
* POST\_TIME — daily local posting time, e.g. 09:00
* TIMEZONE — e.g. Australia/Brisbane
* POST\_MODE — auto (50/50 News/Reddit) or news or reddit

**3) requirements.txt**

requests==2.32.3

praw==7.7.1

tweepy==4.14.0

Pillow==10.4.0

APScheduler==3.10.4

pytz==2024.1

python-dotenv==1.0.1

## 4) Procfile

worker: python main.py

## 5) main.py

import os

import sys

from datetime import datetime

from apscheduler.schedulers.blocking import BlockingScheduler

from apscheduler.triggers.cron import CronTrigger

import pytz

from bot import run\_once

TIMEZONE = os.getenv("TIMEZONE", "Australia/Brisbane")

POST\_TIME = os.getenv("POST\_TIME", "09:00") # HH:MM 24h

def schedule\_job():

tz = pytz.timezone(TIMEZONE)

hour, minute = map(int, POST\_TIME.split(":"))

sched = BlockingScheduler(timezone=tz)

sched.add\_job(run\_once, CronTrigger(hour=hour, minute=minute, timezone=tz))

print(f"[Scheduler] Will post daily at {POST\_TIME} ({TIMEZONE}).", flush=True)

try:

sched.start()

except (KeyboardInterrupt, SystemExit):

print("[Scheduler] Stopped.")

if \_\_name\_\_ == "\_\_main\_\_":

# If RUN\_NOW is set, execute immediately (useful for testing on Railway)

if os.getenv("RUN\_NOW"):

run\_once()

sys.exit(0)

schedule\_job()

## 6) bot.py