

# NotebookLM Briefing-Workflow — Agent Guide v2.0

**Stand:** 2026-07-21 | **CLI:** notebooklm v0.1.0 (npm) | **Autor:** Hector

Vollständige Referenz für AI-Agenten und deren Manager. Deckt Installation, Authentifizierung, alle drei Briefing-Presets (nb1/nb2/nb3) und Troubleshooting ab.

## Inhaltsverzeichnis

- [Toolchain-Installation](#)
- [Authentifizierung](#)
- [CLI-Kommandoreferenz](#)
- [Preset-Übersicht](#)
- [nb1 — Standard-Briefing](#)
- [nb2 — Kurz-Audio zuerst](#)
- [nb3 — Briefing ohne Notebook-Link](#)
- [Routing-Regeln](#)
- [Caption Contracts](#)
- [MP4-Rendering](#)
- [Fehler & Troubleshooting](#)
- [Limitationen v0.1.0 & Workarounds](#)

## 1. Toolchain-Installation

### Voraussetzungen

- Linux-Container (oder macOS/Linux mit Shell-Zugriff)
- Node.js >= 18, npm >= 9
- Kein Root-Zugriff nötig (User-Installation)

### ffmpeg + ffprobe (Static Build)

```
# Download und Installation
FFMPEG_URL="https://github.com/BtbN/FFmpeg-Builds/releases/download/latest/ffmpeg-n7.1-latest-linux64-gpl-7.1.tar.xz"
mkdir -p /tmp/ffmpeg-install
curl -sL "$FFMPEG_URL" | python3 -c "
import lzma, tarfile, io, sys
data = sys.stdin.buffer.read()
with tarfile.open(fileobj=io.BytesIO(lzma.decompress(data))) as tf:
    for member in tf:
        if member.name.endswith(('ffmpeg', 'ffprobe')):
            tf.extract(member, '/tmp/ffmpeg-install')
"
find /tmp/ffmpeg-install -name ffmpeg -exec cp {} ~/.local/bin/ \;
find /tmp/ffmpeg-install -name ffprobe -exec cp {} ~/.local/bin/ \;
chmod +x ~/.local/bin/ffmpeg ~/.local/bin/ffprobe
rm -rf /tmp/ffmpeg-install
```

### yt-dlp (Standalone Binary)

```
curl -sL "https://github.com/yt-dlp/yt-dlp/releases/latest/download/yt-dlp" \
-o ~/.local/bin/yt-dlp && chmod +x ~/.local/bin/yt-dlp
```

### notebooklm CLI (npm)

```
npm install -g notebooklm
```

## Playwright + Chromium (für Login)

```
npm install -g playwright
npx playwright install chromium
```

**Container-Hinweis:** Der Playwright-Headless-Shell (`chrome-headless-shell`) crasht in manchen Container-Umgebungen mit SIGSEGV. Workaround: `/usr/bin/chromium` verwenden (falls vorhanden) oder den Login auf einem Host-Rechner mit Display durchführen.

## PATH setzen

```
export PATH="$HOME/.local/bin:$HOME/.npm-global/bin:$PATH"
# Permanent:
echo 'export PATH="$HOME/.local/bin:$HOME/.npm-global/bin:$PATH"' >> ~/.bashrc
```

## Installation prüfen

```
ffmpeg -version 2>&1 | head -1      # ≥ N-125705
ffprobe -version 2>&1 | head -1
yt-dlp --version                    # ≥ 2026.07.04
notebooklm --version                # 0.1.0
```

---

# 2. Authentifizierung

## Login (einmalig)

```
notebooklm login
```

Öffnet einen Browser. Mit dem gewünschten Google-Konto einloggen. Warten bis die NotebookLM-Startseite sichtbar ist. Dann **Enter** im Terminal drücken.

Das Session-Token wird gespeichert unter: `~/.notebooklm/storage-state.json`

## Headless-Modus (nicht empfohlen):

```
notebooklm login --headless
```

Google OAuth blockt Headless-Logins in der Regel. Nur mit vollem Chromium (nicht Headless-Shell) und viel Glück möglich.

## Fallback-Strategie für Container ohne Display:

Auf einem Host-Rechner mit Browser: `notebooklm login` ausführen  
Die Datei `~/.notebooklm/storage-state.json` in den Container kopieren  
Pfad im Container: `~/.notebooklm/storage-state.json`

## Auth-Status prüfen

```
# Einfacher Test: Notebooks auflisten
notebooklm list
# → Bei Erfolg: Liste der Notebooks (auch leer = ok)
# → Bei Fehler: Auth-Error → Login wiederholen
```

⚠ **v0.1.0 hat keinen auth check Befehl.** notebooklm list ist der einfachste Auth-Test.

## Multi-Account (Fallback für Rate-Limits)

```
# Zweites Konto einrichten
notebooklm login -o ~/.notebooklm/fallback-storage.json

# Bei Rate-Limit: Storage-Pfad tauschen
# (v0.1.0 hat keine NOTEBOOKLM_HOME env var – Storage-Pfad muss
# bei jedem Befehl via -o übergeben werden, was die CLI aber
# nur bei login unterstützt. Workaround: Symlink tauschen.)
```

---

## 3. CLI-Kommandoreferenz

### Basis-Kommandos

```
# Version
notebooklm --version

# Hilfe
notebooklm --help
notebooklm <command> --help
```

### Notebooks

```
# Alle Notebooks auflisten
notebooklm list
notebooklm notebook list

# Notebook erstellen
notebooklm create "Mein Titel"
notebooklm notebook create "Mein Titel"

# Notebook-Info
notebooklm notebook info <notebookId>

# Notebook umbenennen
notebooklm notebook rename <notebookId> "Neuer Titel"

# Notebook löschen
notebooklm notebook delete <notebookId>
```

### Sources (Quellen)

```
# Quellen eines Notebooks auflisten
notebooklm source list <notebookId>

# URL als Quelle hinzufügen (mit Warten auf Verarbeitung)
notebooklm source add -w <notebookId> "<url>"

# Datei als Quelle hochladen
notebooklm source add-file -w <notebookId> "<dateipfad>"

# Text als Quelle hinzufügen
notebooklm source add-text -c "Textinhalt" <notebookId> "Titel"
notebooklm source add-text -f textdatei.txt <notebookId> "Titel"

# Quelle löschen
notebooklm source delete <notebookId> <sourceId>

# Quelle umbenennen
notebooklm source rename <notebookId> <sourceId> "Neuer Titel"
```

```
# Quelle aktualisieren (URL/Drive)
notebooklm source refresh <notebookId> <sourceId>
```

Generate (KI-Inhalte)

```
# Audio Overview (Podcast) generieren
notebooklm generate audio -w -l de -i "Prompt-Text" <notebookId>

# Video Overview generieren
notebooklm generate video -w -l de -i "Prompt-Text" <notebookId>

# Report generieren (Formate: briefing, study-guide, blog)
notebooklm generate report -w -f briefing -p "Prompt" <notebookId>

# Quiz generieren
notebooklm generate quiz -w -n 10 -d medium <notebookId>

# Flashcards generieren
notebooklm generate flashcards -w -n 20 <notebookId>

# Generierungs-Status prüfen
notebooklm generate status <notebookId> <artifactId>
```

Wichtige Flags:

```
-w, --wait — Blockiert bis zur Fertigstellung (sonst läuft Generierung im Hintergrund)
-l, --language <lang> — Sprachcode (de, en, es, fr, ...)
-i, --instructions <text> — Custom Prompt für die Generierung
```

Chat & Research

```
# Frage an ein Notebook stellen
notebooklm ask <notebookId> "Meine Frage"
notebooklm chat ask <notebookId> "Meine Frage"

# Web-Recherche für ein Notebook
notebooklm research web -m deep -i <notebookId> "Suchbegriff"
# -m fast|deep — Recherche-Modus
# -i — Quellen automatisch importieren

# Drive-Recherche
notebooklm research drive -i <notebookId> "Suchbegriff"
```

4. Preset-Übersicht

| Preset | Output                  | Notebook-Link | Audio-Länge              | Infografik |
|--------|-------------------------|---------------|--------------------------|------------|
| nb2    | Kurz-Audio + MP4 + Text | Ja            | Kurz (~7 Min) + Standard | Ja         |
| nb3    | 1 MP4 + Text            | Nein          | Kurz (~7 Min)            | Ja         |

Trigger-Erkennung

```
https://youtube.com/watch?v=XXXXX nb1 → nb1
https://x.com/user/status/123 nb2 → nb2
https://old.bitchute.com/video/XXXXX nb3 → nb3
/pfad/zu/datei.pdf nb1 → nb1
```

Quelle VOR dem Schlüsselwort. Ohne Schlüsselwort → nachfragen.

5. nb1 — Standard-Briefing

Output: 1 Telegram-Nachricht mit MP4-Video (Infografik + Audio) + Text mit allen Links inkl. Notebook-Link.

## Workflow (12 Schritte)

### Schritt 1: Quelle identifizieren

```
# YouTube: Metadaten holen
yt-dlp --print title --print duration "<url>"

# Andere URLs: Titel aus URL extrahieren oder web_fetch
```

### Schritt 2: Auth prüfen

```
notebooklm list
# → Erfolg = auth ok, Fehler = Login nötig
```

### Schritt 3: Notebook erstellen

```
notebooklm create "Quelle/Thema – Kernaussage (2026)"
# → Gibt Notebook-ID aus (z.B. "abc123def456")
```

**Titel-Format:** Quelle/Thema – Kernaussage (Jahr)

**Regel:** Pro Auftrag immer ein NEUES Notebook.

### Schritt 4: Quelle hinzufügen & verarbeiten

```
notebooklm source add -w <notebookId> "<url>"
# -w = warten bis Verarbeitung abgeschlossen
```

### Schritt 5: Audio generieren

```
notebooklm generate audio -w -l de \
-i "Erstelle eine deutsche Audio-Zusammenfassung fuer einen unbekannten allgemeinen Hoerer. Erwahne keine privaten Namen, keinen Auftragge"
<notebookId>
```

### Schritt 6: Infografik generieren

⚠ **v0.1.0-Limitation:** notebooklm generate hat keinen infographic Subcommand. Workarounds:

**Option A:** notebooklm generate report -w -f briefing -p "Infografik-Prompt" <notebookId> — Report als Text-Briefing

**Option B:** Infografik manuell über die NotebookLM-Web-UI erstellen

**Option C:** Infografik durch den Agenten (z.B. via diagram-maker Skill) selbst erstellen lassen

Für den nb1-Workflow: Report als Fallback nutzen, im Caption Contract als "[ ] Briefing-Report" statt "[ ] Infografik" kennzeichnen.

### Schritt 7: Fact-Check

```
notebooklm ask <notebookId> "Nenne die 5 wichtigsten Kernthesen"
notebooklm ask <notebookId> "Was sind die stärksten Belege?"
notebooklm ask <notebookId> "Welche Widersprüche oder offenen Fragen gibt es?"
```

Kernthesen verifizieren. Bei Unstimmigkeiten: User warnieren.

## Schritt 8: Notebook teilen

⚠ **v0.1.0-Limitation:** Kein CLI-Command für share. Workaround:

Über die NotebookLM-Web-UI manuell teilen

Oder: Die API-Methode `share(notebookId, true)` ist im npm-Paket vorhanden, aber nicht als CLI-Kommando exponiert. Kann via Node.js-Script aufgerufen werden:

```
// share-notebook.js
const { NotebookLM } = require('notebooklm');
const nlm = await NotebookLM.fromStorage();
await nlm.share('<notebookId>', true);
const url = await nlm.getShareUrl('<notebookId>');
console.log(url);
```

## Schritt 9: Audio herunterladen

⚠ **v0.1.0-Limitation:** Kein CLI-Command für download. Workaround:

```
// download-audio.js
const { NotebookLM } = require('notebooklm');
const fs = require('fs');
const nlm = await NotebookLM.fromStorage();
const artifacts = await nlm.artifacts.list('<notebookId>', 'audio');
const audioId = artifacts[0].id;
const data = await nlm.artifacts.downloadAudio('<notebookId>', audioId);
fs.writeFileSync('output/audio.mp3', Buffer.from(data));
```

## Schritt 10: MP4 rendern

```
ffmpeg -y \
  -loop 1 -framerate 1 -i infographic.png \
  -i audio.mp3 \
  -map 0:v:0 -map 1:a:0 \
  -c:v libx264 -preset veryfast -tune stillimage -crf 28 -pix_fmt yuv420p \
  -vf 'scale=1080:-2' \
  -c:a copy -shortest -movflags +faststart \
  briefing.mp4
```

**Bei >50MB:** `-crf 32 -vf 'scale=720:-2'`

## Schritt 11: Audio-Dauer ermitteln

```
ffprobe -v error -show_entries format=duration \
  -of default=noprint_wrappers=1:nokey=1 audio.mp3
# → Sekunden in MM:SS umrechnen
```

## Schritt 12: Telegram-Briefing senden

EINE Nachricht mit Text + MP4 als Media. Caption Contract siehe [§9](#).

---

## 6. nb2 — Kurz-Audio zuerst

Wie nb1, aber mit **zusätzlichem Kurz-Audio VOR dem Standard-Briefing**.

Zusätzliche Schritte

Schritt 5b: Kurz-Audio generieren (VOR Standard-Audio)

```
notebooklm generate audio -w -l de \
-i "Erstelle eine KURZE deutsche Audio-Zusammenfassung fuer einen unbekannten allgemeinen Hoerer. Eine Stufe kuerzer als die Standard-Audio
<notebookId>
```

Sendereihenfolge:

**Zuerst** Kurz-Audio als standalone Audio-Nachricht (AUDIO-ONLY, kein MP4)  
**Danach** das normale nb1-Video-Briefing

7. nb3 — Briefing ohne Notebook-Link

Wie nb1, aber:

**Kurz-Audio** statt Standard-Audio (gleicher Prompt wie nb2 Kurz-Audio)  
**Notebook NICHT teilen** — kein Share-Schritt  
**Caption Contract OHNE Notebook-Zeile**  
**Fact-Check optional** (interner Check empfohlen, aber kein Muss)

8. Routing-Regeln

Der Ausgabe-Zielort hängt vom Quell-Chat ab:

| Beauftragt in               | Ausgabe-Ziel                                | Archiv-Ziel    |
|-----------------------------|---------------------------------------------|----------------|
| OME-Gruppe (-1003839640481) | NUR Topic "Notebook Primer" (threadId 7823) | = Ausgabe-Ziel |

**Wichtig:** Bei Beauftragung aus der OME-Gruppe geht das Ergebnis **ausschließlich** ins Topic "Notebook Primer" — nicht an den Sender, nicht in einen anderen Chat.

9. Caption Contracts

nb1 + nb2 (mit Notebook-Link)

```
□ <STARKER TITEL IN CAPS>

<Ein Satz Teaser – warum ist das wichtig, was ist der Kern.>

► Audio-Briefing: <MM:SS Min.>, deutsch
□ Infografik: im Video sichtbar integriert
□ Notebook: <share_url>
□ Originalquelle: <primary_source_url> (<sprachhinweis>)

Warum es zählt: <Ein kompakter Absatz – max. 3 Sätze.>
```

nb3 (ohne Notebook-Link)

□ <STARKER TITEL IN CAPS>

<Ein Satz Teaser – warum ist das wichtig, was ist der Kern.>

- ▶ Audio-Briefing: <MM:SS Min.>, deutsch
- Infografik: im Video sichtbar integriert
- Originalquelle: <primary\_source\_url> (<sprachhinweis>)

Warum es zählt: <Ein kompakter Absatz – max. 3 Sätze.>

## Regeln

Keine privaten Namen im sichtbaren Text  
Keine Tool-Logs, keine Debug-Infos  
YouTube-Links: Original-URL, nicht gekürzt  
Sprachhinweis: (eng.), (de.), (fr.) etc.  
Mehrere Quellen: je eine □ Originalquelle:-Zeile  
**Links als Klartext-URLs** — NICHT als Markdown-Links [text](#) (Telegram rendert das nicht klickbar)  
PDFs: als separate Datei mitschicken

---

## 10. MP4-Rendering

### Standard (<50MB)

```
ffmpeg -y \  
-loop 1 -framerate 1 -i infographic.png \  
-i audio.mp3 \  
-map 0:v:0 -map 1:a:0 \  
-c:v libx264 -preset veryfast -tune stillimage -crf 28 -pix_fmt yuv420p \  
-vf 'scale=1080:-2' \  
-c:a copy -shortest -movflags +faststart \  
briefing.mp4
```

### Komprimiert (>50MB)

```
ffmpeg -y \  
-loop 1 -framerate 1 -i infographic.png \  
-i audio.mp3 \  
-map 0:v:0 -map 1:a:0 \  
-c:v libx264 -preset veryfast -tune stillimage -crf 32 -pix_fmt yuv420p \  
-vf 'scale=720:-2' \  
-c:a copy -shortest -movflags +faststart \  
briefing.mp4
```

### Audio-Dauer ermitteln

```
ffprobe -v error -show_entries format=duration \  
-of default=noprint_wrappers=1:nokey=1 audio.mp3
```

---

## 11. Fehler & Troubleshooting

| Fehler                                       | Ursache                          | Lösung                                                        |
|----------------------------------------------|----------------------------------|---------------------------------------------------------------|
| ffmpeg: not found                            | Static binary fehlt              | Siehe §1 Installation                                         |
| yt-dlp: not found                            | Binary fehlt                     | Siehe §1 Installation                                         |
| Auth-Error bei list                          | Token abgelaufen                 | notebooklm login wiederholen                                  |
| Playwright is not installed                  | playwright fehlt in node_modules | npm install -g playwright && npx playwright install chromium  |
| Headless-Shell SIGSEGV                       | Container-Inkompatibilität       | /usr/bin/chromium verwenden oder Login auf Host-Rechner       |
| generate hängt ewig                          | Kein -w Flag                     | -w für blockierendes Warten, sonst mit generate status pollen |
| MP4 >50MB → Telegram blockiert Video zu groß |                                  | Mit -crf 32 -vf 'scale=720:-2' komprimieren                   |



|                          |                        |                                                          |
|--------------------------|------------------------|----------------------------------------------------------|
| Audio ist M4A trotz .mp3 | NotebookLM liefert M4A | file prüfen, Format ist egal — ffmpeg verarbeitet beides |
| Rate-Limit               | Zu viele Requests      | Fallback-Konto nutzen, 15 Min warten                     |
| Kein Infografik-Command  | v0.1.0-Limitation      | Report als Fallback oder Web-UI                          |
| Kein Download-Command    | v0.1.0-Limitation      | Node.js API-Script (siehe §12)                           |
| Kein Share-Command       | v0.1.0-Limitation      | Node.js API-Script oder Web-UI                           |

---

## 12. Limitationen v0.1.0 & Workarounds

Die npm-Version notebooklm v0.1.0 (von kaelenhou) ist funktional, hat aber Lücken gegenüber der Python notebooklm-mcp-cli:

| Feature              | Python CLI | npm v0.1.0      | Workaround                                        |
|----------------------|------------|-----------------|---------------------------------------------------|
| NOTEBOOKLM_HOME      | ☐          | ☐               | Symmlink für Multi-Account                        |
| source wait          | ☐          | ☐ (via -w Flag) | source add -w                                     |
| artifact wait        | ☐          | ☐               | generate -w blockiert; generate status zum Pollen |
| generate infographic | ☐          | ☐               | generate report -f briefing oder Web-UI           |
| download audio       | ☐          | ☐               | Node.js API-Script                                |
| share public         | ☐          | ☐               | Node.js API-Script oder Web-UI                    |
| artifact list        | ☐          | ☐               | Node.js API-Script                                |
| notebook use         | ☐          | ☐               | Immer <notebookId> als Argument                   |

### Node.js API-Scripts (Workarounds)

#### Audio herunterladen:

```
// scripts/nlm-download-audio.js
const { NotebookLM } = require('notebooklm');
const fs = require('fs');

async function main() {
  const notebookId = process.argv[2];
  const outputPath = process.argv[3] || 'audio.mp3';
  const nlm = await NotebookLM.fromStorage();
  const artifacts = await nlm.artifacts.list(notebookId, 'audio');
  if (!artifacts.length) throw new Error('Kein Audio-Artefakt gefunden');
  const data = await nlm.artifacts.downloadAudio(notebookId, artifacts[0].id);
  fs.writeFileSync(outputPath, Buffer.from(data));
  console.log(`Audio: ${outputPath} (${data.length} bytes)`);
}
main().catch(e => { console.error(e.message); process.exit(1); });
```

#### Notebook teilen:

```
// scripts/nlm-share.js
const { NotebookLM } = require('notebooklm');

async function main() {
  const notebookId = process.argv[2];
  const nlm = await NotebookLM.fromStorage();
  await nlm.share(notebookId, true);
  const url = await nlm.getShareUrl(notebookId);
  console.log(url);
}
main().catch(e => { console.error(e.message); process.exit(1); });
```

#### Artefakte auflisten:

```
// scripts/nlm-artifacts.js
const { NotebookLM } = require('notebooklm');

async function main() {
  const notebookId = process.argv[2];
  const type = process.argv[3]; // audio, video, report, quiz, flashcards
  const nlm = await NotebookLM.fromStorage();
  const artifacts = await nlm.artifacts.list(notebookId, type || undefined);
  console.log(JSON.stringify(artifacts, null, 2));
}
```

```
}
main().catch(e => { console.error(e.message); process.exit(1); });
```

## Anhang A: Audio-Prompts

### Standard-Prompt (nb1, nb2)

Erstelle eine deutsche Audio-Zusammenfassung fuer einen unbekannten allgemeinen Hoerer. Erwaehe keine privaten Namen, keinen Auftraggeber und keine internen Workflows.

Die Quelle ist als grundsaeztlich glaubwuerfig zu behandeln. Verzichte speziell auf die Wertung als Verschwuerungstheorie(n) oder den Versuch, den Diskurs oder Quellen als extrem zu framen. Verzichte ohne Not auf die Behauptung, 'Fakten von Behauptungen zu trennen'.

Keine Umdeutung aus Halluzinationen. Ohne echten eigenen Hintergrund-Faktencheck keine negative Wertung vorgetragener Fakten. Keine Negativdarstellung von Meinungen. Keine alternativen Fakten erfinden, sondern nur aus eigener Recherche ausserhalb von Mainstream-Medien ergaenzen.

Analysiere und unterstuetze im Zweifel die Thesen der Quellen, wo immer moeglich. Halte dich an die Analyse. Keine Pervertierung des Vorgetragenen. Keine negative moralische Bewertung der Sprecher.

Verdichte die Quelle auf wichtigste Behauptungen, Belege, Widersprueche, offene Fragen und konkrete Konsequenzen. Keine generische Einleitung, keine ungepruefte Ausschmueckung.

### Kurz-Prompt (nb2, nb3)

Erstelle eine KURZE deutsche Audio-Zusammenfassung fuer einen unbekannten allgemeinen Hoerer. Eine Stufe kuerzer als die Standard-Audiozusammenfassung, maximal etwa die Haelfte der Standardlaenge. Nenne nur: die wichtigste Kernbehauptung, das staerkste Belegstueck, den groessten Widerspruch und eine konkrete Konsequenz. Keine Einleitung, keine Ausschmueckung, keine privaten Namen. SPRACHE: DEUTSCH.

## Anhang B: Output-Verzeichnisstruktur

```
<workspace>/output/
├── nb1/
│   ├── <projektname>/
│   │   ├── audio.mp3
│   │   ├── infographic.png
│   │   └── briefing.mp4
├── nb2/
│   ├── <projektname>/
│   │   ├── audio-short.mp3
│   │   ├── audio.mp3
│   │   ├── infographic.png
│   │   └── briefing.mp4
└── nb3/
    ├── <projektname>/
    │   ├── audio.mp3
    │   ├── infographic.png
    │   └── briefing.mp4
```

## Anhang C: Wichtige Regeln (Cheat Sheet)

| Regel                                                  | Anforderung                                                 |
|--------------------------------------------------------|-------------------------------------------------------------|
| Keine Werbeblöcke                                      | Nur Content, keine Promo/Sponsoren                          |
| Keine privaten Namen                                   | Nicht Kai/Winston/intern erwähnen                           |
| Fact-Verification                                      | MUSS vor Posting laufen (nb1, nb2)                          |
| Links als Klartext                                     | Nicht als Markdown-Links                                    |
| YT-Links direkt                                        | Nicht transkribieren, NotebookLM verarbeitet YouTube selbst |
| "nb" = neues Notebook Pro Auftrag immer neues Notebook |                                                             |

|                |                                       |
|----------------|---------------------------------------|
| Archiv-Kopie   | Jedes Artifact MUSS ins Archiv        |
| Eine Nachricht | Text + Media zusammen, nie getrennt   |
| -w Flag        | Immer für blockierendes Warten nutzen |
| -l de          | Sprache immer auf Deutsch setzen      |
| -i "prompt"    | Custom Instructions immer mitschicken |

**Version:** 2.0 | **Datum:** 2026-07-21 | **CLI:** notebooklm v0.1.0 (npm) | **Autor:** Hector

Diese Anleitung basiert auf der tatsächlich installierten Toolchain im Container bot-hector.

Skills (notebooklm-core, notebooklm-pipelines, notebooklm-agent-guide) wurden für die Python-CLI geschrieben und sind für die npm-CLI angepasst.

NotebookLM Briefing-Workflow — Agent Guide v2.0 | 2026-07-21 | CLI: notebooklm v0.1.0 (npm) | Hector

Erstellt für AI-Agenten und deren Manager. Interne Referenz — nicht zur Veröffentlichung.