

SHIFT AI WEBINAR — GAS入門⑤

Slackに自動で 通知が届くようにする60分

Incoming Webhookを覚える／中級者向け・今日も「見て学ぶ」回です



けいたろう

業務自動化エキスパート

SHIFT AI WEBINAR GAS入門⑤

2026.10.14 (水)

本シリーズについて

01

月2回・年内全10回

GAS入門シリーズ。毎回ちがう「自動化の型」を1つずつ覚えていきます。

02

単発参加OK

各回は独立した内容。第1～4回に出ていなくても、今日だけで完結します。

03

今日は第5回

前回のDiscordに続き、**Slack**へ送ります。型は同じ、違うのは「投函口の取り方」だけ。

今日は中級者向け・Slackを使う人向け——次のページの事前テストで確かめます

今日の進め方

今日は——

わたしが実際に作るところを 見ていただく回です

コードと手順書は、いまチャットに貼ったURLにぜんぶあります。一緒にやりたい方はそこから。
追いつけなくて大丈夫。ご自身の実装は後日でかまいません。

このあと事前テスト（5問）で点数を出し、チャットに書いてもらいます

事前テスト——5問中3つ以上で「中級」（受講資格あり）

- 1 GASのエディタ（Apps Script）を自分で開いたことがある
- 2 コードを貼り付けて▶実行したことがある
- 3 承認ダイアログ（「詳細」→「移動」→「許可」）を通したことがある
- 4 トリガー（時計のアイコン）を作ったか消したことがある
- 5 「Webhook」の意味を、人にひとことで説明できる

3～5点 = 中級

今日の想定レベルです。どんどんコメントしてください。
改善点も積極的に反映します

0～2点

参加は歓迎です。今日は見ただけで「こんなことができるんだ」を持ち帰ってください。第1～3回のアーカイブで基礎をやると、次は3点以上になります

「やったことがある」の数があなたの点数——次のページでチャットに書いてください

いま、チャットに点数を書いてください（1分）

書き方は「数字だけ」

- 例: 4 / 2 / 0
- 3点以上の方: 今日はコメント大歓迎。つまりき・質問・改善点をどんどん
- 2点以下の方: 見るだけでOK。コメントは控えめに（3点以上の方の質問を優先して拾います）

この回が向かない方（正直に言います）

- Slackを使っていない・使う予定がない方には、今日の内容は意味がありません
- 「なぜSlackなのか」が疑問の方は、今回は聞かないほうがいいです（第4回 Discord／第6回 LINE へ）。「何ができるか見たい」だけなら流し見で歓迎

今日は中級者向け・Slackを使う人向け——何度でも念押しします

チャット・リアクション、大歓迎です

01

なんでもチャットへ

気になったこと、聞き逃したこと、どんなことでも気軽にどうぞ。

02

理解チェックは数字だけ

1=OK／2=なんとなく／3=置いていかれた。文章にしなくて大丈夫です。

03

アーカイブがあります

あとから見返せます。その場で全部わからなくて大丈夫です。

チャットの質問は**セクションの切れ目**で拾います

PROFILE

自己紹介



けいたろう

業務自動化エキスパート／書籍著者。

GoogleのツールとGAS（Google Apps Script）で「手作業をなくす」業務自動化をテーマに、セミナー登壇や書籍の執筆をしています。

本シリーズでは、毎回1時間で自動化の型を1つずつお見せします。



PART 1

今日の前提 と SlackのWebhook

用意するもの、そして「投函口」のもらい方を実画面で（9分）／念押し: 今日の中級者向け（事前テスト3点以上）です

用意するもの／要らないもの

必要なもの

- **Googleアカウント**（Gmailが使えていればOK）
- **Slackのワークスペース**（無料・自分だけの練習用でOK）
- Slackの**Webhook URL**（取り方はこのあと実画面で）

要らないもの

- インストール作業
- クレジットカード・課金
- プログラミングの経験

会社のSlackより自分の練習用ワークスペースが安全——なぜ: 会社では管理者の承認が必要がある

Webhookとは？——Slackは「アプリ」を先に作ります

前回 (Discord)

- Webhook = **合言葉付きの投函口** (専用URL)
- チャンネルの設定から**すぐ**もらえた
- 投函口のURLを知っていれば誰でも投稿できる → **人に見せない**

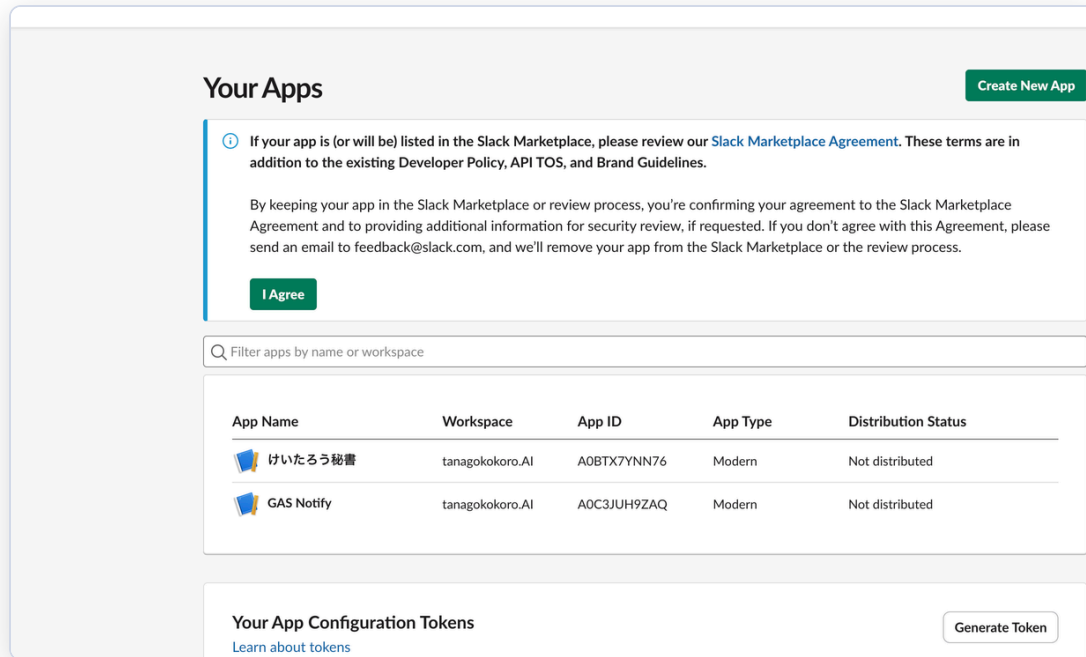
今日 (Slack)

- 投函口の考え方は**まったく同じ**
- ただし **「アプリ」**を1つ作ってから投函口をもらう
- なぜ: Slackは **「誰が投稿するか」**を先に決める仕組みだから。投稿はそのアプリの名前で届く

違いは投函口のもらい方だけ——送り方は前回と同じ

Webhook URLの取り方（前半）——アプリを作る

1 api.slack.com/apps → 右上の Create New App



Your Apps [Create New App](#)

ⓘ If your app is (or will be) listed in the Slack Marketplace, please review our [Slack Marketplace Agreement](#). These terms are in addition to the existing Developer Policy, API TOS, and Brand Guidelines.

By keeping your app in the Slack Marketplace or review process, you're confirming your agreement to the Slack Marketplace Agreement and to providing additional information for security review, if requested. If you don't agree with this Agreement, please send an email to feedback@slack.com, and we'll remove your app from the Slack Marketplace or the review process.

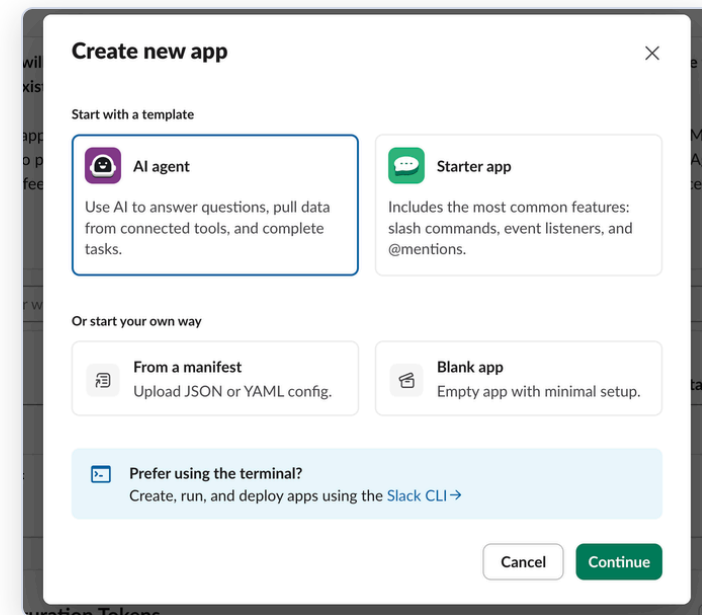
[I Agree](#)

🔍 Filter apps by name or workspace

App Name	Workspace	App ID	App Type	Distribution Status
けいたろう秘書	tanagokoro.AI	A0BTX7YNN76	Modern	Not distributed
GAS Notify	tanagokoro.AI	A0C3JUH9ZAQ	Modern	Not distributed

Your App Configuration Tokens [Learn about tokens](#) [Generate Token](#)

2 Blank app を選んで Continue → 名前（英数字）とワークスペース → Create



Create new app ✕

Start with a template

AI agent
Use AI to answer questions, pull data from connected tools, and complete tasks.

Starter app
Includes the most common features: slash commands, event listeners, and @mentions.

Or start your own way

From a manifest
Upload JSON or YAML config.

Blank app
Empty app with minimal setup.

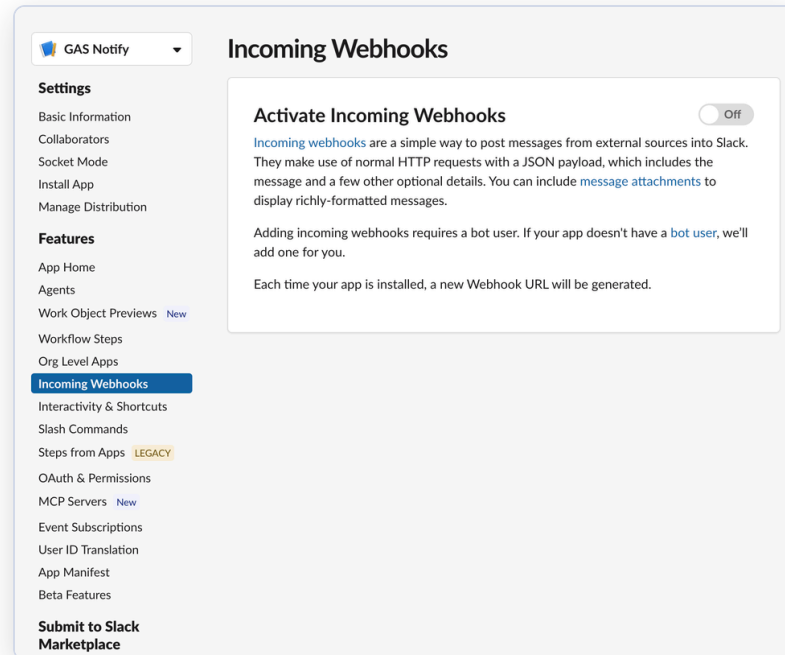
Prefer using the terminal?
Create, run, and deploy apps using the [Slack CLI](#) →

[Cancel](#) [Continue](#)

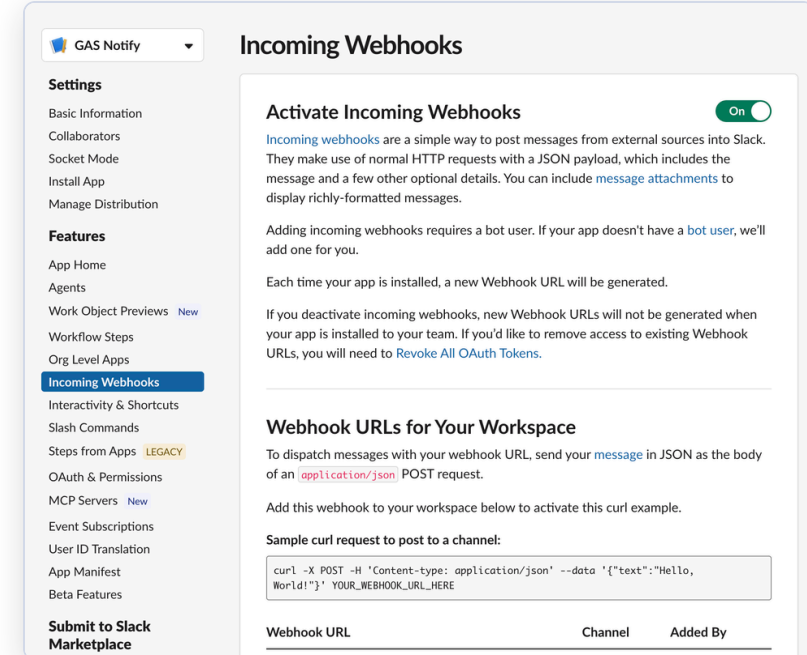
名前は英数字だけ（日本語は弾かれます）——例: GAS Notify

Webhook URLの取り方（後半A）——機能をONにする

3 左メニュー Features の中の Incoming Webhooks



4 右上のスイッチを Off → On（画面が下に伸びる）



なぜ？——Slackのアプリは機能を1つずつONにする作り。今日使うのは Incoming Webhooks だけ

Webhook URLの取り方（後半B）——投函口をもらう

5 画面の一番下 Add New Webhook

App Home
Agents
Work Object Previews New
Workflow Steps
Org Level Apps
Incoming Webhooks
Interactivity & Shortcuts
Slash Commands
Steps from Apps LEGACY
OAuth & Permissions
MCP Servers New
Event Subscriptions
User ID Translation
App Manifest
Beta Features
Submit to Slack Marketplace
Review & Submit
Give feedback
Slack ❤️

Each time your app is installed, a new Webhook URL will be generated.

If you deactivate incoming webhooks, new Webhook URLs will not be generated when your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to [Revoke All OAuth Tokens](#).

Webhook URLs for Your Workspace

To dispatch messages with your webhook URL, send your **message** in JSON as the body of an **application/json** POST request.

Add this webhook to your workspace below to activate this curl example.

Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text": "Hello, World!"}' YOUR_WEBHOOK_URL_HERE
```

Webhook URL	Channel	Added By
No webhooks have been added yet.		

Add New Webhook

6 日本語の許可画面 → チャンネルを選んで「許可する」

Review app permissions

Slack へのアクセスを「GAS Notify」アプリに許可する

このアプリは、参加中のワークスペース tanagokokoro.AI のメンバーにより作成されました。

Slack はあなたに付与されている権限を GAS Notify と共有します。Slack の [プライバシーポリシー](#) を参照してください。あなたに付与されている権限が Slack によって共有されることに同意すると、GAS Notify のプライバシー規約とサービス利用規約にも同意したものとみなされます。インストールする前に GAS Notify の規約を確認してください。

Workspace
tanagokokoro.AI

Webhook 用のチャンネル
チャンネルを検索する...

"GAS Notify" がアプリとして投稿することのできるチャンネルを指定する必要があります。

キャンセル **許可する**

許可すると元の画面に**URLが1行**できる → Copy。それが投函口（人に見せない）

CHECK 1 / 4

ここまで理解できましたか？

ここまでの3行

- Webhook=合言葉付きの投函口。Slackはアプリを作ってからもらう
- 押すのは5か所: **Create New App → Incoming Webhooks On → Add New Webhook → 許可 → Copy**
- URLは人に見せない（手順書ステップ0に同じ画面）

チャットに数字を1つ

- 1 OK、進んで大丈夫**
- 2 なんとなく**
手順書があれば自分でもできそう
- 3 置いていかれた**
画面をもう一度

10秒待ちます。3が多ければ、前の2枚を指しながら「押すのは5か所」と言い直します。多数決で進みます。

PART 2

GAS超入門

2

3分だけ。毎回やる定番パートです（前回参加の方は復習）

GASとは？——「外へ送れる」道具です

01

Googleと直結

カレンダー・Gmail・スプレッドシートを、面倒な認証なしにそのまま操作できます。

02

トリガーで自動実行

「時刻が来たら」「フォームが届いたら」——PCを開いていなくても動きます（第1～3回）。

03

外のサービスへも送れる

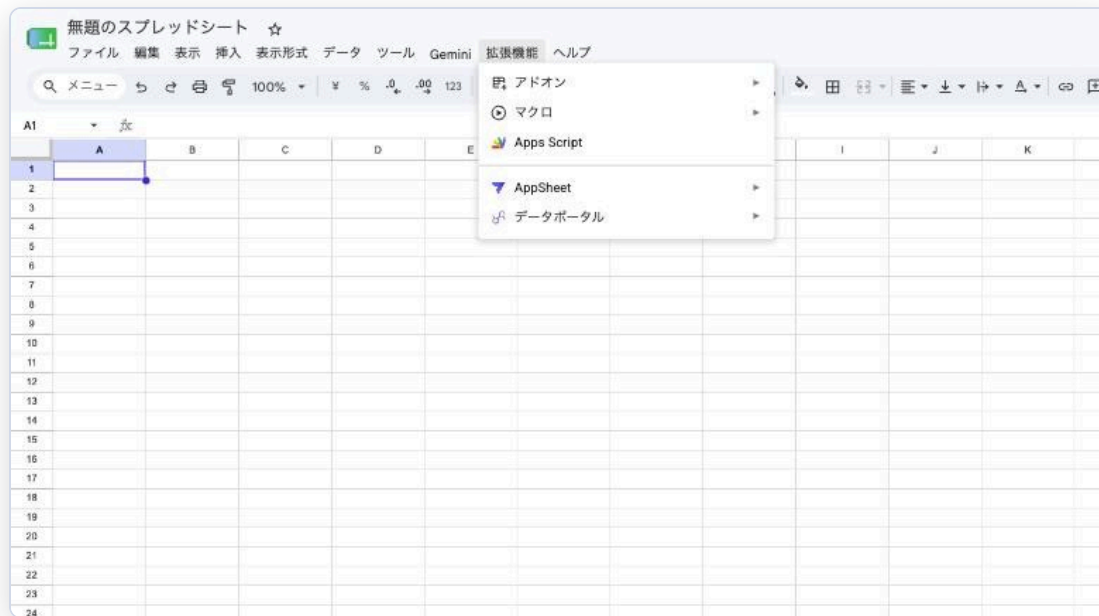
URLに向かって文章を投げられます。
Discord（前回）・**Slack（今日）**・LINE（次回）。

今日はスプレッドシートを使います——送った記録（ログ）をシートに貯めるため。第1・2回と同じ開き方

HOW TO OPEN

エディタの開き方——今日はスプレッドシートから

1 新しいスプレッドシート → メニュー「拡張機能」→「Apps Script」



2 新しいタブでエディタが開く（ここにコードを貼る）



なぜシートから？——送った記録を「ログ」シートに1行ずつ残すため。第3・4回はドライブから単体で作りました

PART 3

今日作るもの

仕組みを1枚の図で。動くところは実演で初めてお見せします（5分）／念押し: 今日では中級者向け（事前テスト3点以上）です

仕組みの全体像——前回と違うのは2点だけ

毎朝7時、今日の予定がSlackに届く。やっていることは前回と同じ3つです

TRIGGER

毎朝7時に起動

時間主導トリガーが、決まった時刻にスクリプトを呼び出します。

CALENDAR

今日の予定を取る

カレンダーから、その日の予定をまとめて取得します。

WEBHOOK

Slackへ投げる

違い①: 送り先が**Slack**のURL。違い②: JSONのキーが **content** → **text**。
送った記録はシートに1行残す。

送り先と1単語を変えるだけ——これがWebhookの型を覚えた人の強み

CHECK 2 / 4

ここまで理解できましたか？

ここまでの3行

- 作るのは「毎朝、今日の予定がSlackに届く」ツール
- やることは**起動する・取ってくる・投げる**の3つ（前回と同じ）
- 前回と違うのは**送り先URL**と**content→text**の2点だけ

チャットに数字を1つ

1 OK、実演に進んで大丈夫

2 なんとなく
動くところを見れば分かりそう

3 置いていかれた
図をもう一度

3が多ければ、前の図を指しながら「送り先と1単語だけです」と言い直します。

4

PART 4

ここから実演します

3ステップ・20分（ターミナル → GAS → 毎朝）。見ているだけでOKです／念押し: 中級者向け（3点以上）のペースで進めます

ターミナルとは？——文字でPCに命令する窓

Macの方

- アプリ名は「**ターミナル**」
- 開き方: **Spotlight** (⌘+スペース) に「ターミナル」と打つ／Launchpad → 「その他」
- 黒い（または白い）窓が開いて、**文字を打つ場所**が出ればOK

Windowsの方

- アプリ名は「**PowerShell**」（「コマンドプロンプト」でも可）
- 開き方: **スタートボタン**で「powershell」と打つ → Enter
- 青い（または黒い）窓が開いて、**文字を打つ場所**が出ればOK

なぜ挟む？——Webhookは「URLに投げるだけ」。いちばん小さな道具で1回体感すると、GASが何をしているか分かる

STEP 1

ステップ1：ターミナルから「Hello」を1回送る

Mac — ターミナルに1行貼って Enter

curl

```
curl -X POST \  
  -H 'Content-type: application/json' \  
  --data '{"text":"Hello from terminal!"}' \  
  https://hooks.slack.com/services/ (あなたのURL)
```

Windows — PowerShell に1行貼って Enter

curl.exe

```
curl.exe -X POST ^  
  -H "Content-type: application/json" ^  
  -d "{\"text\":\"Hello from terminal!\"}" ^  
  https://hooks.slack.com/services/ (あなたのURL)
```

- **curl** = 「このURLにこれを投げて」と頼む道具。Mac・Windowsに最初から入っている
- Slackに「**Hello from terminal!**」が届いたら成功。**これがWebhookの全部**
- なぜ英語？——ターミナルは**日本語の文字化け**が起きやすいから。日本語はGASで送ります
- **URLの行は画面に映さない**（人に見せない）

「**URLにJSONをPOSTする**」を、道具を変えて2回見る——次はGASで同じことをします

STEP 2

ステップ2：GASから送る＋ログをシートに残す

sendHello → sendToSlack — ターミナルと同じことをGASで `code.gs`

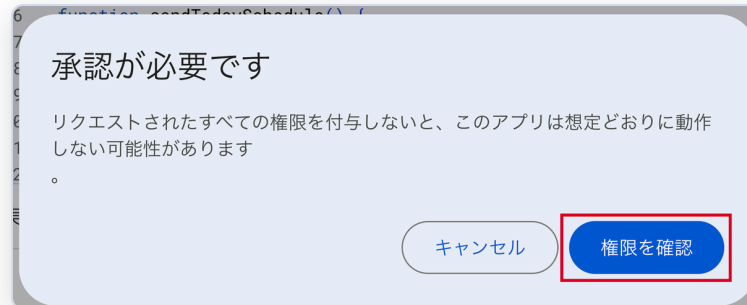
```
const WEBHOOK_URL = ' (コピーしたURL) ';
function sendHello() {
  sendToSlack('こんにちは！GASからの初投稿です');
}
function sendToSlack(text) {
  UrlFetchApp.fetch(WEBHOOK_URL, { // ① このURLに
    method: 'post',                // ③ POSTで
    contentType: 'application/json',
    payload: JSON.stringify(
      { text: text }                // ② Slackは text
    )
  });
  logToSheet(text);                // ④ シートに1行残す
}
```

- **UrlFetchApp.fetch()** = さっきの **curl** と同じ。URLに向かって送る部品
- 送る中身は **{ text: "文章" }** の1行だけ（Discordは content でした）
- **logToSheet()** = 日時・本文・結果を「**ログ**」シートに**1行**追記。なぜ: あとで「いつ・何を送ったか」を見返せる
- **URLを貼るとき、画面を隠します**（人に見せない）

ターミナルでやったことを**GASに引っ越す**だけ——届いたら、シートにも1行増えている

ここで必ず一度止まります

1 「権限を確認」



2 左下の「詳細」



3 中身を見て許可

無題のプロジェクト がアクセスできる情報を選択してください

☒ すべて選択

 Gmail のすべてのメールの閲覧、作成、送信、完全な削除 です。 [アクセスの詳細を表示](#) ☒

 Google カレンダーを使用してアクセスできるすべてのカレンダーの表示、編集、共有、完全な削除 です。 [アクセスの詳細を表示](#) ☒

 自分がいなくてこのアプリケーションを実行できるようにします です。 [アクセスの詳細を表示](#) ☒

なぜ出る？——自分で書いたコードに自分で許可を出すため。それだけです

CHECK 3 / 4

ここまで理解できましたか？

ここまでの3行

- ターミナルの **curl** で1回、GASの **sendHello** で1回——**同じURLに同じJSON**を投げた
- ▶実行 → 承認ダイアログを「**詳細**」→「**移動**」→「**許可**」で通した
- Slackに届き、「**ログ**」シートに**1行増えた**——これが今日の核

チャットに数字を1つ

1 OK、次のステップへ

2 なんとなく
手順書があれば自分でもできそう

3 置いていかれた
承認のところをもう一度

3が多ければ、承認ダイアログと「届いた」画面だけをもう一度ゆっくり見せます。

STEP 3

ステップ3：毎朝の予定を、Slackへ

送り先を Discord → Slack に変えただけ

code.gs

```
function sendTodayScheduleToSlack() {  
  const today = new Date();  
  const events = CalendarApp.getDefaultCalendar()  
    .getEventsForDay(today);           // 今日の予定を取る  
  sendToSlack(                          // ← ステップ1の部品  
    buildMessage(today, events));  
}  
function createDailyTrigger() {  
  deleteDailyTrigger();                // 二重登録を防ぐ  
  ScriptApp.newTrigger('sendTodayScheduleToSlack')  
    .timeBased().atHour(7)  
    .everyDays(1).create();             // 毎日7時台  
}
```

- 予定を取るのは第3回・第4回と**まったく同じ**
- 最後の1行だけ **sendToSlack()** に差し替え
(ログも自動で残る)
- **createDailyTrigger** は**1回だけ**実行すればOK
- なぜ1回？——トリガーは**登録しておく仕掛け**だから。毎回実行するものではない

部品を1つ差し替えるだけで、送り先が変わる——Webhookの型

動いているかの確認と、止め方（2通り）

左メニューの「トリガー（時計のアイコン）」がコントロールパネルです。

- **確認**: sendTodayScheduleToSlack の行が1件あればON ／ **止める①**: 「:」 → 「トリガーを削除」
- **止める②**: Slack側で Incoming Webhooks の**ゴミ箱アイコン**（またはスイッチをOff）。**時刻は分単位で指定できません**（7時～8時のあいだ）



GAS側でもSlack側でも止められる——戻せると分かれば安心して試せる

CHECK 4 / 4

ここまで理解できましたか？

ここまでの3行

- 予定を取る部分は前回と同じ。**送る部品だけ差し替えた**
- **createDailyTrigger** を**1回実行**すれば、毎朝動く
- 止め方は**GAS側・Slack側の2通り**

チャットに数字を1つ

1 OK、流れはつかめた

2 なんとなく
後日、手順書を見ながらやってみる

3 置いていかれた
トリガーのところをもう一度

3が多ければ、トリガー画面を見せて「1回実行しただけ」を強調します。ここが最後のチェックです。

うまく動かないとき トップ3

症状	よくある原因	直し方
「URLを貼ってください」と出る	URLを貼り替えていない／引用符を消した	'https://hooks.slack.com/services/...'の形になっているか確認
実行数に 403 / 404 / no_service	URLが欠けている／Slack側でOff・削除済み	Incoming Webhooks の画面でもう一度Copyして貼り直す。スイッチがOnか確認
「こんにちは」は届くのに予定が届かない	カレンダーの承認が未完了／別のカレンダー	sendTodayScheduleToSlack をもう一度実行し、承認を最後まで進める

ほとんどはこの3つ——手順書にも同じ表を載せています

この型でできること

01

チャンネルを送り分ける

Add New Webhook でURLを増やすだけ。「営業」「総務」に別々の予定を流せます。<!channel> で全員に通知も。

02

AIに要約させてから送る

予定の一覧をGeminiに渡して「今日の一言」を付けてから投稿。ただ通知するだけで終わらない使い方です。

03

次回はLINEへ

「URLにJSONをPOST」は同じ型。第6回はLINE公式アカウントに送ります。**事前準備が重い回**なので告知が出たら早めに。

投函口の取り方さえ覚えれば、送り先はいくらでも増やせる

アンケートのお願い——「事前テストの点数」は必ず

1

**冒頭の事前テストは何点でしたか
——必ず回答**

0～5点。点数を踏まえて意見を読みます。3点以上の方の改善点は積極的に反映します

2

悪かった点は、その点数を前提に理由を明確に

「速かった」ならどこから。「分かりにくかった」ならどの画面・どの言葉か。改善します

3

前回の声はこう反映しました

画面共有の直し方・Slack実画面の追加・「なぜ」の説明・事前テスト。今回の声も次回に

褒め言葉より点数付きの具体的な不満が、次の回を良くします

まとめ／次回・Q&A

Webhookの型は同じ——違うのは「投函口の取り方」だけ

TAKEAWAY 01

Slackは「アプリ」を作ってから投函口（Webhook URL）をもらう。押すのは5か所

TAKEAWAY 02

コードと手順書は配布します。実装は後日でOK

NEXT

次回・第6回はLINE連携。準備が重いので、告知が出たら早めに

次回の日程は、SHIFT AI会員サイトのイベントページでご案内します。

本日はありがとうございました。質問はチャットへどうぞ——答えきれない分は次回の冒頭で拾います。

アンケートでは事前テストの点数を必ず、悪かった点は理由までぜひ。次回までに改善します。