

次世代のSlack App開発 「new Slack platform」について

@stanigome

new Slack platformとは

Slack App を開発するための新しいプラットフォーム

Denoで開発出来る

- 型がしっかりサポートされている
- 導入が簡単

SlackのCloud上で動くので自分でサーバーを立てる必要が無い

専用のCLIも使うことでセットアップからデプロイまでが簡単

Easy x Fast x Secure

Denoとは？

- TypeScriptを標準サポート
- node_modulesが無くnpm installがいらぬ
- Node.jsの作者が開発



セットアップ















CLIを使う

```
// 雛形の作成
slack create my-app

// Triggerを設定
slack trigger create --trigger-def "triggers/greeting_trigger.ts"

// ローカル起動
slack run

// デプロイ
slack deploy
```

- >  .github/workflows
- >  .vscode
- >  assets
- >  datastores
- >  functions
- >  triggers
- >  workflows
-  .gitignore
-  LICENSE
-  README.md
-  deno.jsonc
-  import_map.json
-  manifest.ts
-  slack.json

構成

Functions

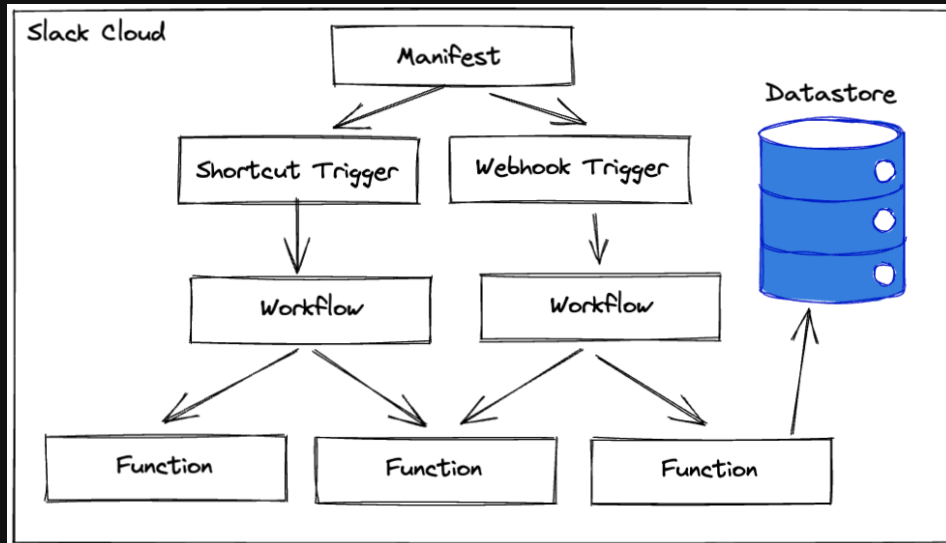
Workflows

Triggers

Datastore

Manifest

これらのパーツを組み合わせてアプリを作っていきます。



Functions

様々なアクションをFunctionとして切り出します。

例: メッセージを送信する、ユーザーを取得する、など

メッセージを送信する

```
export const FunctionDefinition = DefineFunction({
  callback_id: "function",
  title: "Post message function",
  source_file: "src/functions/function.ts",
  input_parameters: {
    properties: {
      userId: {
        type: Schema.slack.types.user_id,
      }
    },
    required: ["userId"],
  },
});

export default SlackFunction(
  FunctionDefinition,
  async ({ inputs, token }) => {
    const client = SlackAPI(token);
    client.chat.postMessage({
      channel: inputs.userId,
      text: "Hello!",
    });
    return {};
  }
);
```

Workflows

Functionsを組み合わせ、実行する順番だったりを定義します。

Functionからのoutputを次のFunctionのinputとして渡すことができます。

build-in Functionsもあります。

<https://api.slack.com/future/functions>

ユーザーからメンションを受け取り、そのメッセージをパースして返信する

```
const Workflow = DefineWorkflow({
  callback_id: "workflow",
  title: "workflow",
  input_parameters: {
    properties: {
      ...
    }
  },
  required: ["userId", "message"],
});
// メッセージを受け取ってパースする
const parsed = Workflow.addStep(ParseFunctionDefinition, {
  message: Workflow.inputs.message
});
// パースしたメッセージを送信する
Workflow.addStep(Schema.slack.functions.SendDm, {
  user_id: Workflow.inputs.userId,
  message: parsed.outputs.text,
});
```

Triggers

Workflowsが実行される条件を定義します。

Link Triggers

発行されたURLをクリックするとWorkflowが実行されます。

Scheduled Triggers

毎日、毎週、毎月などの周期で定期実行されます。

Event Triggers

メンション、リアクションなどのイベントが発生したときに実行されます。

Webhook Triggers

発行されたURLに対してPOSTするとWorkflowが実行されます。

Link Trigger

```
const LinkTrigger: Trigger<typeof Workflow.definition> = {
  name: "Sample App",
  type: "shortcut",
  workflow: "#/workflows/workflow",
  inputs: {
    userId: {
      value: "{{data.user_id}}",
    },
  },
};
```

Event Trigger

```
const EventTrigger: Trigger<typeof Workflow.definition> = {
  name: "Sample App",
  type: "event",
  workflow: "#/workflows/workflow",
  event: {
    event_type: "slack#/events/app_mentioned",
  }
};
```

Datastore

Cloud上にDatastoreが用意されています。

`put`, `get`, `query` の3つの操作ができます。

DynamoDB syntaxが使えるので複雑な条件での検索も可能です。

Datastoreの定義

```
export const Datastore = DefineDatastore({
  name: 'datastore',
  primary_key: "id",
  attributes: {
    id: {
      type: Schema.slack.types.user_id,
    },
    foo: {
      type: Schema.types.string,
    },
  },
});
```

データの取得

```
const getDatastore = await client.apps.datastore.get<
  typeof Datastore.definition
>({
  datastore: 'datastore',
  id: inputs.userId,
});
```

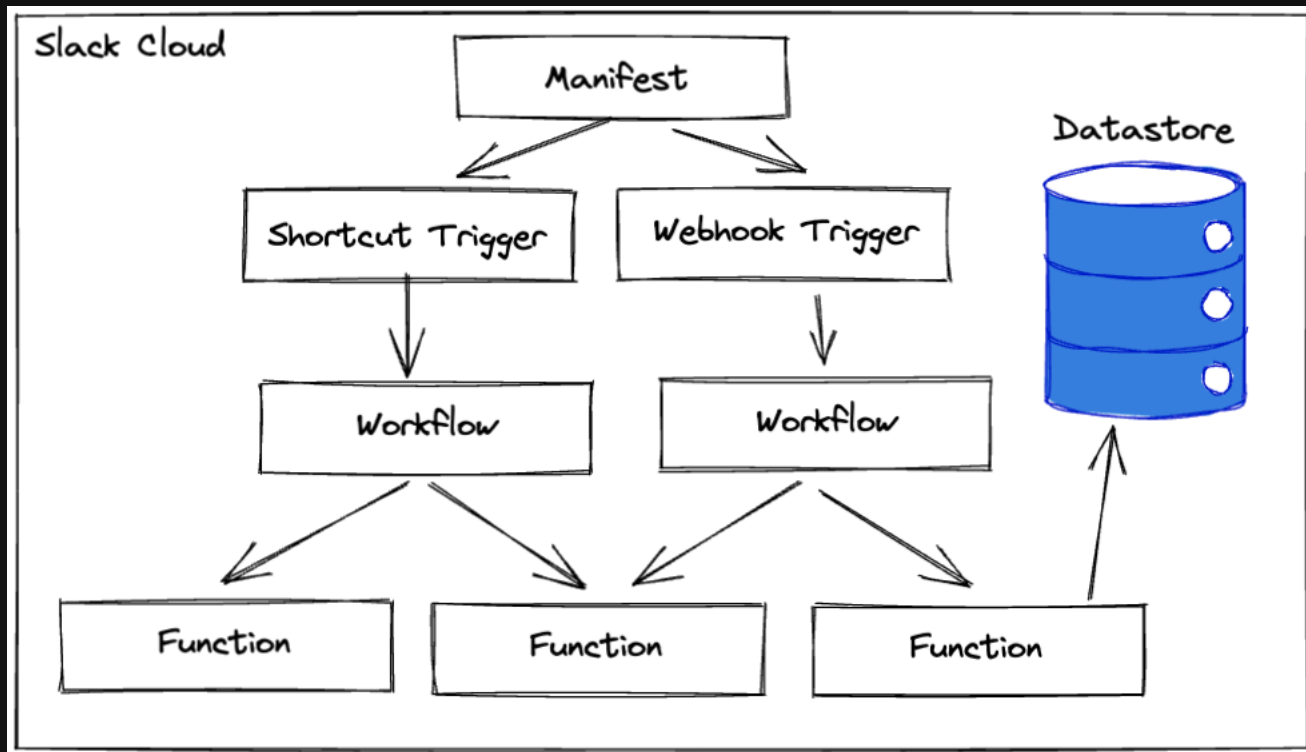

Manifest

アプリの設定を定義します。

今までWeb上で行っていたものがコードで設定できるのでとても便利です。

```
const definition: SlackManifestType = {
  runOnSlack: true,
  name: "sample bot",
  description: "",
  icon: "assets/icon.png",
  workflows: [Workflow, TokenWorkflow],
  types: [MessageType],
  datastores: [MessageDatastore],
  botScopes: [
    "commands",
    "chat:write",
    "chat:write.public"
  ]
};
export default Manifest(definition);
```

組み合わせる




実際の例

「匿名で意見を投稿出来るアプリ」を作ってみます。

流れ

1. リンクをクリックしてフォームを開く
2. フォームを入力する
3. Botがチャンネルにその内容を投稿する

 **Form** ×

匿名で投稿されます

☐ アイデア

☐ 不満

☐ その他

内容

Write something

Cancel

Submit

Link Trigger

失敗談

リアクションや返信をしていないメッセージ(見逃したメッセージ)を再通知するbotを作ったが・・・



rereinder-bot (dev) APP 14:03

from: @Naoya Ishii in: #p_taishosky_dev

Resolve

@geek_murata 🍷 Kyosuke Takeda 🍷 @stanigome

お疲れ様です。

掲載者も加えた公開が来週26日（時間帯は午前中予定、調整中）に決まりました。

そこで本番反映までの懸念点洗い出しやタスク確認のMT...

[message link](#)

from: @tgt_kminegishi in: #p_jds_niihamasmart

Resolve

👤 Daichi Tsukada 👤 JDSさんに連絡して提案をすすめていただけますでしょうか。

@stanigome 谷米さん ありがとうございます！

[message link](#)

from: @stanigome in: #stanigome_playground

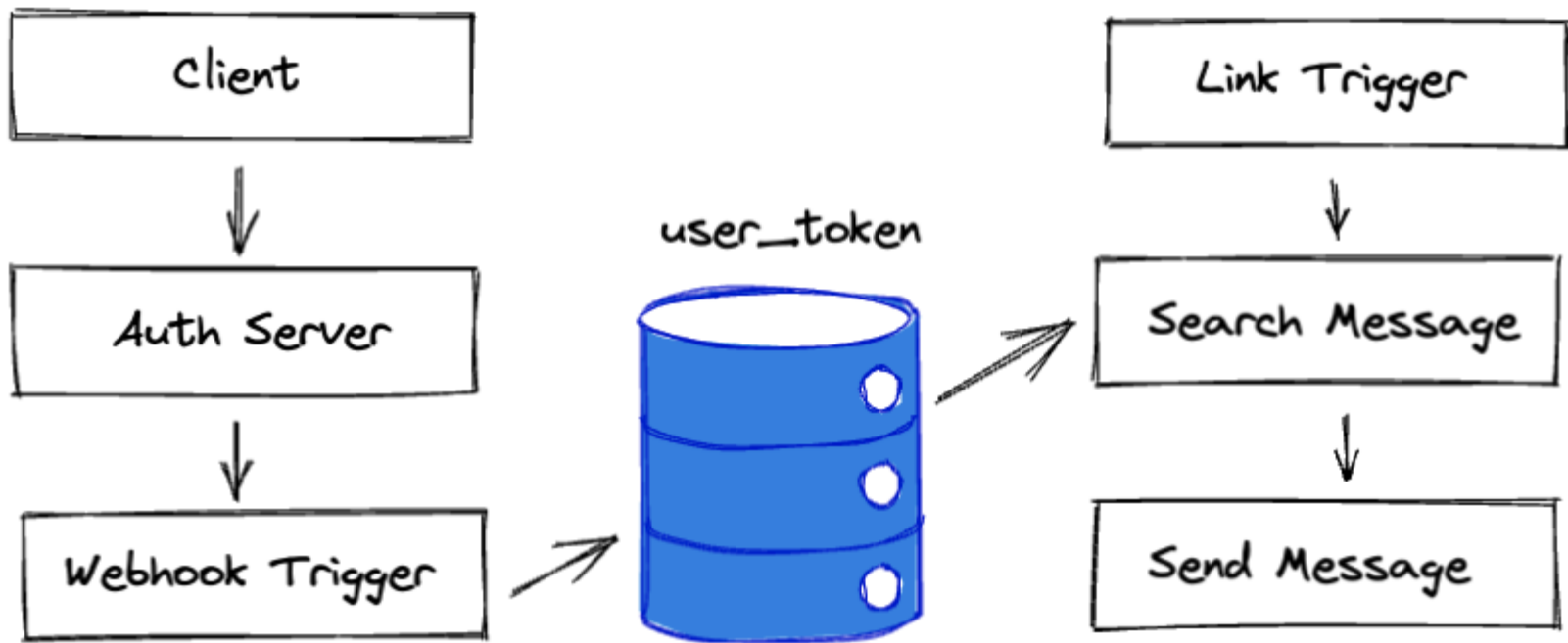
Resolve

@stanigome

aiueo

[message link](#)

構成



new Slack platform では、まだ、出来ないこと

OAuth2認証をしてuser_tokenを取得する処理をCloud内だけでは完結出来ない(認証サーバーを別途立てる必要がある)

manifestにredirectUrlsを設定するとデプロイ出来ない(ローカルでは開発出来る)

CLIからデプロイしたアプリは手動で設定することが出来ない

- Client Idを取得できないのでそもそもOAuth2認証が出来ない
- redirectUrlsを手動で設定することも出来ない

→user_tokenを使用したアプリは現状作ることが出来ない😞

| | | | |
|--|---------------|-----------------|---|
|  rereminder-bot Created with Slack CLI | TSUNAGU GROUP | N/A |  |
|  rereminder-bot (dev) | TSUNAGU GROUP | Not distributed | |

```
const definition: SlackManifestType = {
  runOnSlack: true,
  name: "rereminder-bot",
  description: "",
  icon: "assets/icon.png",
  workflows: [Workflow, TokenWorkflow],
  types: [MessageType],
  datastores: [MessageDatastore, TokenDatastore],
  features: {
    appHome: {
      messagesTabEnabled: true,
      messagesTabReadOnlyEnabled: true,
    },
  },
  // @ts-ignore
  redirectUrls: [env.REDIRECT_URL],
  botScopes: [
    "commands",
    "chat:write",
    "chat:write.public",
    "datastore:read",
    "datastore:write",
  ],
  userScopes: [
```

まとめ

まだβ版なので出来ないこともあります、今後も開発が進んでいくと思います。

とても簡単にSlackアプリを作ることが出来るので、

アイデアがある人やDenoを触ってみたい人はぜひ作ってみてください！