

# Slack Workflow Project Structure

Slack automation workflow with custom functions, triggers, and workflow steps using the Slack CLI.

#slack #workflow #automation #typescript #deno

PNG

PDF

Copy

Prompt

## Project Directory

### my-workflow/

```
manifest.ts App manifest an...
slack.json Slack CLI confi...
import_map.json Deno import map...
> functions/ Custom functions
  > my_function/
    definition.ts Function schema
    handler.ts Function logic
    handler_test.ts
  > send_notification/
    definition.ts
    handler.ts
> workflows/ Workflow defini...
  my_workflow.ts Main workflow
  onboarding_workflow.ts
> triggers/ Workflow trigge...
  link_trigger.ts Shareable link
  scheduled_trigger.ts Time-based
  event_trigger.ts Slack events
> datastores/ Data storage
  my_datastore.ts Datastore schema
> types/ Custom types
  my_custom_type.ts
README.md
.gitignore
LICENSE
```

## Why This Structure?

Slack's new automation platform uses Deno and TypeScript. Workflows chain functions together, and triggers start workflows. Functions are reusable building blocks with defined inputs/outputs. Datastores provide persistent storage within Slack.

## Key Directories

**manifest.ts** - App definition, scopes, and features  
**functions/** - Reusable custom functions with handlers  
**workflows/** - Workflow definitions chaining functions  
**triggers/** - How workflows are started (link, schedule, event)

## </> Function Definition

```
// functions/greeting/definition.ts
import { DefineFunction, Schema } from "deno-slack-sdk/mod...

export const GreetingFunction = DefineFunction({
  callback_id: "greeting_function",
  title: "Generate Greeting",
  source_file: "functions/greeting/handler.ts",
  input_parameters: {
    properties: {
      user_id: { type: Schema.slack.types.user_id },
    },
    required: ["user_id"],
  },
  output_parameters: {
    properties: {
      greeting: { type: Schema.types.string },
    },
    required: ["greeting"],
  },
});
```

## > Getting Started

1. Install Slack CLI: `curl -fsSL https://downloads.slack-edge.com/slack-cli/install.sh | bash`
2. `slack create my-workflow`
3. `slack run` for local development
4. Create trigger: `slack trigger create --trigger-def triggers/link_trigger.ts`
5. `slack deploy` to production

## ☑ When To Use This

- Automating repetitive Slack tasks
- Custom approval workflows
- Scheduled notifications and reminders
- Data collection from team members
- Integration between Slack and external services

## ⚡ Trigger Types

**Link triggers** - Shareable URLs that start workflows  
**Scheduled** - Run on a schedule (daily, weekly)  
**Event** - React to Slack events (reaction added, message)  
**Webhook** - External systems can trigger workflows

## ☑ Best Practices

- Keep functions small and focused
- Use datastores for persistent data, not external DBs
- Test handlers with `deno test`
- Use link triggers for user-initiated workflows
- Handle errors gracefully with proper output

## ⚖ Trade-offs

**Deno runtime** - Different from Node.js, limited npm support  
**Slack-hosted** - Runs on Slack infrastructure, limited customization  
**Paid plan required** - Workflows require Slack paid plans