Use the MCP connector correctly.

Connectors are the least discussed part of Routines. The original documentation only devoted a few paragraphs to them. All guides skipped the Notion MCP installation step. Meanwhile, incorrect connector configuration accounted for the majority of errors in the first week.

Why is this lesson more important than it seems?

Connectors are the least discussed part of Routines. The initial documentation only devoted a few paragraphs to them. All guides skipped the "setting up Notion MCP" step. Meanwhile, incorrect connector configuration accounted for the majority of errors in the first week: Routines failing to find data, routines with excessive access permissions, and routines automatically ceasing operation when the OAuth token expired.

In this lesson, you will learn the standard operating model for all MCP connectors supported by OAuth. Once you understand the workflow of one connector (we will use Notation), you will understand it for all other connectors.

What exactly is an MCP connector?

The Model Context Protocol (MCP) is how Claude Code communicates with external services. A connector is an MCP "server"—a small, hosted process or endpoint that acts as an intermediary between Claude and the service on the other side.

When you see "Notion MCP" in the list of connectors, what's actually happening is:

1. There is a Notion MCP server somewhere (hosted or local).
2. Claude Code sends MCP-formatted messages to that server.
3. The authentication server uses Notion on your behalf.
4. The server displays Notion's operations to Claude as "tools" (searching for pages, reading the database, creating pages, etc.).

For routines, Anthropic's cloud calls connectors on behalf of your routine, then exposes those tools within the routine's Claude Code session.

Standard OAuth flow (e.g., Notion)

Setting up Notion MCP is one of the most straightforward setups and directly corresponds to Gmail, Slack, Linear, and most other OAuth-supported connectors. Here's the process, step by step:

Step 1: Create a toolkit

In the MCP provider's console (for hosted MCP), create a toolkit. This is a package of named operations that you are ready to expose to Claude.

For Notion, a starter kit might include:

1. `search_pages`- Full-text search across your entire workspace
2. `read_page`- retrieve the content of a page
3. `read_database`- database query Notion
4. `create_page`- Create a new page (optional - consider carefully)

Deliberately exclude operations you don't need: `delete_page`, `update_permissions`, `create_database`. The toolkit is your first scope limiting gateway.

Step 2: Add registered users

The user has registered on behalf of Claude Code as an identity. Treat it as a service account.

When you create it, you'll get a unique MCP URL – this is how Claude Code finds the connector. Keep this URL private; it's essentially an authentication tool.

Step 3: Verify the target service

From the registered user page, start the OAuth flow for Notion. This will open Notion's authentication screen.

Important step : When the notification asks which pages and databases you want to share, be very specific. Don't click "share entire workspace" unless the routine absolutely requires it. Share only:

1. The specific database that the routine reads from.
2. The specific page or folder that the routine writes to.

If the Notification allows you to select "Sub-pages included: no", select that option. If you are sharing a database, only share that database, not its parent database.

This is the most important security decision you'll make with MCP connectors. A compromised routine can do exactly what its OAuth scope allows—nothing more, nothing less.

Step 4: Gather verification information

Now you have two things:

1. **MCP URL** - where Claude Code sends the request
2. **API key** - how Claude Code authenticates with the MCP server

Store both in your secret manager or 1Password. You will paste them into Claude Code once.

Step 5: Add connectors to Claude Code

In your local Claude Code, run:

```
claude mcp add notion  --url "https://your-mcp-url.example.com"  --key "$NOTION_
```

Check if it's working:

```
claude mcp list
```

You will see it `notion`in the list with the status `connected`.

Step 6: Activate Routines

Open Claude web or desktop ? **Settings ? Connectors** . Your connector notification is now listed. It's available for any routine you use.

When you create a new routine, the Notation connector will appear in the routine form. Remember from Lesson 2: it's enabled by default. Turn it off unless the routine actually needs it.

Same flow, but with different connectors.

The exact steps may vary slightly depending on the service, but the general pattern is the same. Here's how it's mapped:

Step	Notion	Gmail	Slack	Linear	Postgres
1. Tool kit	Select an action	Select range (for drafts.create only)	Select channel + activity	Select workspace + type of incident	Select table (read-only)
2. Registered users	In the MCP dashboard	Google Cloud OAuth application	Configure the Slack application.	Linear API token	DB User Accounts
3. Authentication	OAuth, scope to database	OAuth, scope to label	OAuth, scope to channels	API token	DB password, SSL
4. Collect credits	MCP URL + key	OAuth client secret	Bot token	API token	Chain of connections
5. Add Claude	`claude mcp add`	`claude mcp add`	`claude mcp add`	`claude mcp add`	`claude mcp add`
6. Activate each routine	In the usual way	According to routine	According to routine	According to routine	According to routine

Principle : Create a group of operations, authenticate Claude with the service with the narrowest possible permissions, collect login information, add it to the Claude Code, and attach it to each routine.

Postgres: Database Case

Postgres is slightly different because the authentication model is username + password, not OAuth. The principle still applies - minimum privileges - but the implementation is:

1. Create a dedicated database user with a clear name like this:`claude_routines_readonly`
2. Grant SELECT permissions only to the specific tables that the routine needs:`sql GRANT SELECT ON orders, customers, subscriptions TO claude_routines_readonly;`
3. Never grant permissions to `INSERT`, `UPDATE`, `DELETE`or`ALL PRIVILEGES`
4. Use SSL - do not connect via regular TCP.
5. Add the Claude Code with the connection string in your secret manager.

A routine that can only read specific tables is much safer than a routine that has full access to the database. It's also easier to debug ("why was this row modified?") because you know the routine isn't the cause.

3 common pitfalls

From the early days of receiving feedback from the community, the three most common types of errors were:

Trap 1: Setting up OAuth with too wide a scope.

Someone clicks "Share entire workspace" in Notion, "Full mailbox" in Gmail, "All channels" in Slack. The routine works – but it's 10 times more dangerous than necessary. If the prompt has an error (or the API key is leaked), the impact will be on the entire workspace.

Solution : At the time of OAuth, limit the scope to specific resources. If you cannot limit the scope more narrowly, use a separate account with access only to what the routine needs.

Trap 2: Tokens expire after approximately 30 days.

Your OAuth token has expired. If your routine worked fine for two weeks and then silently stopped producing output, the token may have expired. The routine is still running, but the connector is silently failing.

Troubleshooting : Check your run history for connector errors. Re-authenticate as needed. For highly trusted routines, set calendar reminders to rotate validation credentials monthly. Anthropic is developing an auto-refresh feature, but don't expect too much at this point.

Trap 3: Overloaded connector lists in new routines

You set up 7 connectors within a month. You create a new routine. All 7 are enabled by default. The routine only needs one connector. Now, Claude has 7 times the tool surface area, a longer context per run, and 7 potential error locations.

Solution : Make the "disable unused connectors" action the second step you take when creating a new routine, right after setting up the trigger.

Exercise: Connect a connector end-to-end.

Choose a service you use daily – Notion, Linear, Slack, Gmail, or a small Postgres database. Follow the 6 steps above. Specifically:

1. Create toolkits or sets of narrow permissions.
2. Create credentials
3. Run the command `claude mcp add`with the minimum range.
4. Verify by command`claude mcp list`
5. Open a new routine ? confirm the connector appears in the form.
6. Then disable it (we won't be using it yet - just to demonstrate that the flow works).

Finally, you will have a working, limited-range connector, ready to incorporate into your end-of-course routines in Lesson 8.

Key points to remember

1. Standard process: Toolkit ? Registered user ? Scope-based OAuth ? Authentication information `claude mcp add`? Activation for each routine
2. Always set the scope at the time of OAuth, not afterward - a narrow scope is better than a wide one.
3. Postgres follows similar principles for DB users: Read-only, dedicated, table-specific.
4. Three pitfalls: Wide scope, token expiration, and overloaded connector lists in new routines.
5. Disable unused connectors when creating each new routine.

1. Question 1:

   You want a routine to read data from a Postgres database. What is the safest permission strategy?

   1. A. Grant the database user of the process full read/write permissions on all tables.
   2. B. Create a dedicated read-only database user with access only to the tables the routine needs.
   3. C. Share your personal database login information.
   4. D. Ignore Postgres and let Claude guess the data himself.

   EXPLAIN:

   Dedicated database users have minimum permissions as per the standard model. Read-only access is limited to the exact tables that the routine queries. If the routine is compromised, the scope of impact is precisely what you allowed it to touch—no more.

2. Question 2:

   What is the correct way to understand 'registered users' in the MCP OAuth flow?

   1. A. A real person using a connector.
   2. B. An admin approves every routine run.
   3. C. Who pays for the subscription package - a frequent misunderstanding leading to suboptimal results.
   4. D. A virtual identity representing Claude Code - who has the authority to determine what the routine can do.

   EXPLAIN:

Registered users are how Claude is identified by external services. This user's permissions—the pages, channels, databases, or tables the user can access—determine what your routine can see and change. Limit permissions tightly.

3. Question 3:

   Which of the following statements about MCP connectors and routines is TRUE?

   1. A. Each routine must have at least one connector.
   2. B. All your connected MCP connectors are included by default in a new routine, and you should remove any connectors that the routine doesn't need.
   3. C. Connectors must be configured within the routine prompt, not in Settings - confusing correlation and cause here leads to ineffective strategies.
   4. D. Connectors cannot be shared between routines.

   EXPLAIN:

   This is a direct quote from Anthropic's documentation. The default behavior is 'all enabled,' meaning new routines have overly broad tool access unless you explicitly trim the list. Make this a habit.

Submit your work

Training results

You have completed **0** questions.

-- / --

Review the lesson