Tools Playbook: n8n, Make, Zapier, Replit, Claude and Codex

This playbook answers one question for each tool: **what do I do first, what do I ask for help, and where do I check the facts?** It is written so you can use it without any video. It uses fake data only.

> **Honesty note.** Tools change often. The steps below are stable habits. Feature names, plans and prices must be checked on the tool's own website the day you act. Community posts (for example Reddit) are useful for ideas, but check them against the official documentation before you rely on them.

1. One rule for every tool

1. **Write the goal in one sentence.**
2. **Use fake data and a test account.**
3. **Build one small step, test it, then add the next.**
4. **Ask for approval before anything is sent, deleted or changed for real.**

2. Choosing a tool for the job

Job	Start with	Why
Connect apps on a trigger, with branches and an approval step	n8n or Make	Visual flows, routing and error paths
Quick connections between popular apps, no code	Zapier	Large app library and simple setup
Build and host a small web app	Replit (or Lovable)	Code, preview and deploy in one place
Write, review or change code in a project	Claude Code or Codex	Agents that read files, run commands and propose changes
Think, write and plan	Claude or ChatGPT	Chat with projects, skills or instructions

3. How to ask for help (any tool)

Good questions get good answers. Give the AI these five things:

Part	Example
Tool and version	"n8n, self-hosted, latest I can see in the editor"
Goal in one line	"When a fake hire is posted, add one row to a test sheet."
What you did	"Webhook node, test URL, Sheets append node."
What you see	"The Sheets node shows an error: 'Column not found'."
What you want	"Tell me which field is wrong and the exact expression to fix it."

4. n8n: first steps and common errors

1. Create a workflow, add a **Webhook** node (POST, test URL), and click Listen for test event.
2. Send one fake body. Open the JSON view. Data sits under `body`.
3. Add a Sheets node. Map `{{ $json.body.hire_id }}` to a column.
4. Activate the workflow when the test passes, and switch the form to the production URL.

Error	Likely fix
"undefined" in a field	Check the path in the JSON view; add `body`
Nothing happens on a real event	Workflow not active, or the test URL is in use
Credential error	Reconnect the test account; check the permissions

5. Make: first steps and common errors

1. Create a scenario and add a trigger module (for example Google Forms: Watch Responses).
2. Run once after one fake submission. Check the bundle.
3. Add the action module and map the fields. Run once again.
4. Add a router with a fallback path, then turn the schedule on.

Error	Likely fix
Scenario is on but nothing runs	Check the trigger and schedule settings
Bundle disappears	Router with no matching filter and no fallback

Operations used up quickly	Lengthen the schedule; check the usage page

6. Zapier: build your first automation (fake data)

Zapier connects apps. A **Zap** is one automation. It has one **trigger** (the event that starts it) and one or more **actions** (what happens next).

6.1 Words you need

Word	Meaning
Zap	One automation: trigger plus actions.
Trigger	The event that starts the Zap, such as a new form response.
Action	A step that does something, such as add a row or send a message.
Filter	A rule that lets a run continue only if a condition is true.
Path	A branch that runs different actions for different conditions.
Test step	A run that uses real sample data from the app, so you can check the fields.

6.2 Build a first Zap: form to sheet

1. **Create a test form** with two fake fields (for example name and rating). Submit one fake response.
2. **Create a Zap.** Choose the form app as the trigger, then choose "New response".
3. **Test the trigger.** Zapier should find your fake response. If it does not, submit another and test again.
4. **Add an action** to a spreadsheet app: "Create spreadsheet row". Map each form field to a column.
5. **Test the action.** Check the test sheet has one new row with the right values.
6. **Publish (turn on) the Zap** only after the test passes.

Pass when: one fake response creates one row, with each value in the right column.

6.3 Add a filter and a second step (safe checks)

1. Add a **Filter** step after the trigger: continue only if the rating is 3 or more.
2. Submit one fake response with rating 5 and one with rating 1.
3. **Pass when:** only the rating 5 response reaches the sheet, and the rating 1 response stops at the filter.

6.4 Before you send any message

- Use a test channel or your own test address. Never send to a real customer while building.
- Add a human approval step before any message that leaves your organisation.
- Check the account each app is connected to. A personal account in a shared Zap is a common mistake.

6.5 Common errors and fixes

Symptom	Likely cause	What to do
Trigger test finds nothing	No sample item exists yet	Create a fake item in the source app, then test again
Action fails on a field	A required field is empty	Map a value to every required field, or give a default
Duplicate rows	The same item triggers twice	Store the item's ID and filter out IDs you have already seen
Zap runs but nothing appears	A filter stopped the run	Open the run history and read which step stopped it
Connection expired	The app login was revoked	Reconnect the test account and test again

6.6 Zapier checklist

Item	Done
Test account and fake data only	[ ]
Trigger and action tested with a sample item	[ ]
Filter tested with one passing and one failing item	[ ]
No live message without an approval step	[ ]
Run history reviewed after the first ten runs	[ ]

Plans, task limits and app availability change. Check zapier.com (help and pricing pages) on the day you build.

7. Replit: first steps and common errors

1. Create a Repl from a template and press Run. Read the first red line in the console if it fails.
2. Store any API key in Secrets. Never paste it into code or chat.
3. Add one route, test it, then deploy and test the deployed address.

Error	Likely fix
Module not found	Add the package in the Packages panel and run again
Works in preview, fails deployed	Check the run command and secrets in deployment

Full steps are in the Replit guide in this series.

8. Claude: getting the most from it

- **Projects:** put standing instructions (tone, audience, rules) in the project once, instead of retyping them.
- **Skills:** save a repeated task as a skill with a header, rules, examples and a checklist. Use it again and again. See the Skills guide.
- **Context:** start a new chat when the topic changes. Check the context panel with `/context` in Claude Code.
- **Claude Code:** keep a `CLAUDE.md` file with the project's standing rules, so every session starts with them.

9. Codex: a safe first session

Codex is an AI coding agent. It can read a project, propose changes, and run commands in it. Treat it like a new team member: give it a small, clear task, and review every change.

9.1 Words you need

Word	Meaning
Agent	A tool that works in steps: it reads, plans, edits and runs commands.
Diff	The exact lines a change adds or removes. Always read the diff before accepting.
Sandbox	A limited area where the agent can run commands safely.
Approval	Your yes or no before the agent runs a command or changes a file.

9.2 Your first session (fake practice project)

1. **Make a copy** of a small practice project. Never start in your main code.
2. **Ask for a plan first.** Example: "Explain what you would change to add a 'hello' page. Do not edit files yet."
3. **Approve one step at a time.** Read each proposed diff before you accept it.
4. **Run the project's tests** yourself after each accepted change.
5. **Pass when:** the tests pass, and every changed line is one you understand.

9.3 Keep it safe

Risk	Control
Secrets in code	Keep keys in environment settings; never paste a key into a prompt
Changes you do not understand	Ask for an explanation of the diff before you accept it
Commands that delete or publish	Refuse them unless you have reviewed the exact command
Work in your main project	Use a copy or a separate branch, and review before you merge

9.4 Good requests and bad requests

Good	Bad
"Add input validation to the form on this page, and add one test."	"Make the app better."
"Explain the error on line 12 before changing anything."	"Fix everything."
"List the files you would change, then wait for my OK."	"Rewrite the whole project."

9.5 Common problems

Problem	Fix
The change breaks the tests	Ask for the failing test output, then one fix at a time
The agent edits too many files	Ask it to stop and list the files, then narrow the task
You cannot tell what changed	Ask for a plain-English summary of the diff, file by file

9.6 Codex checklist

Item	Done
Practice copy used, not the main project	[ ]
Plan requested before any edit	[ ]
Every diff read before accepting	[ ]
Tests run after each change	[ ]
No secrets in the project or the prompt	[ ]

Check the current Codex documentation from OpenAI on the day you use it. Access, permissions and plan limits change.

10. Where to learn more (check each site on the day)

Tool	Start here	Good for
n8n	docs.n8n.io	Nodes, webhooks, AI agents, hosting
Make	help.make.com	Scenarios, routers, data stores, AI
Zapier	zapier.com/help	Zaps, filters, app connections
Replit	docs.replit.com	Projects, secrets, deployment
Claude	docs.anthropic.com	Models, tools, skills, agents
Codex	OpenAI developer documentation	Agents that work on code

Community posts (for example Reddit discussions) can show real problems people hit. Use them to find questions to check, not as the final answer.

11. Ask-the-AI checklist

Before you ask	Done
I wrote the goal in one sentence	[ ]
I used fake data	[ ]
I included the exact error message	[ ]
I asked for one fix at a time	[ ]