

A neighbourhood concierge bot

Make a bot answer the right question and show its limits clearly.

Build

Check

Try independently

The situation and your goal

Situation

Residents ask your fictional bot about the next meetup. It should answer confirmed facts, show help and acknowledge unknown commands. It must not promise to be a human organizer.

Keep this artifact

A tested bot, a command table and a webhook troubleshooting note.

Understand the mechanism

01

A bot has an event boundary

A Telegram message creates an update. A trigger receives it, a workflow chooses a reply and the Telegram API sends the response. A text prompt alone is not a deployed bot.

02

One webhook needs a clear owner

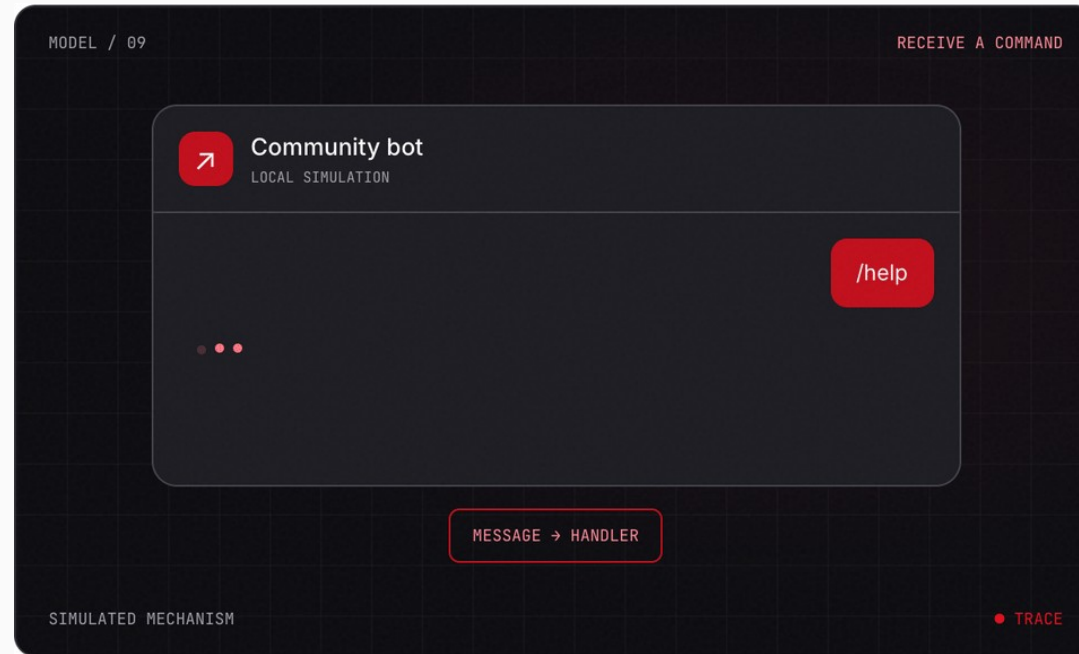
Test and production configurations can compete for a bot webhook. Use a dedicated test bot and understand which workflow currently receives its updates.

03

Unknown commands need a route

A resident should get useful help when a command is absent or unsupported. Silence and invented answers both make a bot hard to trust.

Explore before you build



Create a test identity

- 1 Use BotFather to create a dedicated test bot. Store the token only in n8n credentials. Write an intro saying this is an automated demo.
- 2 Check: The bot has a clear identity and the token is not in slides or public files.
- 3 Hint: Use a separate test bot from an existing community bot.

Prepare the command table

- 1 Define /start, /help and /next plus a fallback. The /next response uses only the approved Saturday10:00 community-room brief.
- 2 Check: Every supported command has one expected response.
- 3 Hint: Do not add unknown perks or live facts.

Configure delivery

- 1 Add Telegram Trigger and connect credentials in n8n. Confirm that the webhook URL is public HTTPS from the trigger execution environment.
- 2 Check: Your own test message reaches the correct workflow.
- 3 Hint: A laptop localhost URL cannot be reached by Telegram without a suitable public webhook setup.

Route and reply

- 1 Use explicit command conditions or a Switch node. Connect Telegram Send Message to the received chat ID.
- 2 Check: Each command replies only to the chat that sent the test request.
- 3 Hint: Do not hardcode a public group ID for this exercise.

Test failure cases

1

Send unknown text, a missing command and /next. Check the execution log. If delivery stops, inspect test/production webhook ownership.

2

Check: Fallback and confirmed facts work; logs show input and chosen route.

3

Hint: Telegram bots cannot generally start a private chat with someone who has never contacted them.

Hand off the bot

- 1 Document commands, token rotation location, active workflow, fallback and the approval step needed before adding it to a community group.
- 2 Check: Another organizer can troubleshoot one failed update.
- 3 Hint: Do not include the token in the handoff document.

When it goes sideways

01

The workflow receives no updates

Webhook URL is unreachable or another workflow owns it. Try: Check public HTTPS and use one active test/production owner.

02

Replies reach the wrong chat

A chat ID was hardcoded. Try: Map the chat ID from the current trigger update.

03

The bot invents answers

An unbounded AI step replaced a known command table. Try: Use explicit approved responses first and retain an unknown-answer path.

Review your evidence

- 1 Dedicated test bot and private credential storage are used.
- 2 Four command routes including fallback pass.
- 3 Reply chat ID comes from the incoming update.

Review your evidence

- 1 Public webhook and ownership are documented.
- 2 Automated identity is clear.
- 3 No unauthorized group broadcast occurs.

What identifies the reply destination?

1

A hardcoded group used for every request.

2

The chat ID on the incoming update.

3

The bot name alone.

Why use a dedicated test bot?

- 1 To avoid webhook conflicts with an existing live bot.
- 2 To skip credentials.
- 3 To send messages to anyone without consent.

Build with a clear contract

Design a Telegram test-bot command table for /start, /help, /next and unknown text. Approved /next facts: Saturday 10:00, community room, free entry. Identify the bot as an automated demo. Explain the n8n Telegram Trigger to routing to Send Message flow, mapping reply chat ID from the update, public HTTPS webhook requirements, test/production webhook conflicts and private token storage. Keep unsupported answers unknown.

[Open copyable prompt](#)

Add a source-backed FAQ

1

Every FAQ has a source.

2

Missing answers are acknowledged.

3

An organizer can revoke or update a fact.

Build independently, then teach it back

Your next challenge

Add two more approved FAQ entries and an unknown-answer route. Then compare a fixed table with an optional retrieval-based draft.

Explain your decisions

Show your artifact, one test that passed, one failure you repaired and one boundary the system respects.