Telegram Bot + Cloudflare Worker Setup Guide

1. Register BotFather Bot

BotFather is Telegram's official bot management tool. All Telegram bots must be created and managed through BotFather.

This step creates your bot and provides the API token, which is required for your Cloudflare Worker to communicate with Telegram.

1.1 Open BotFather

1. Open Telegram (mobile or desktop)
2. Search for @BotFather
3. Select the verified account (blue checkmark)
4. Tap Start or send /start

1.2 Create a New Bot (/newbot)

1. Send /newbot
2. Enter a bot display name (shown to users)
3. Enter a unique bot username ending with 'bot'

Rules:

- Username must end with 'bot'
- Username must be unique
- Usernames are case-insensitive

1.3 Receive and Secure the API Token

BotFather will return an API token.

Security notes:

- Do not share this token
- Do not commit it to source control
- Store it securely as TELEGRAM_BOT_TOKEN in Cloudflare

1.4 Optional Bot Configuration

Optional BotFather commands:
/setdescription – Bot profile description
/setabouttext – Short about text
/setuserpic – Bot profile picture

1.5 Verify Bot Creation

1. Click the bot username link
2. Open the bot chat
3. Confirm the Start button appears

2. Create Cloudflare Worker

A Cloudflare Worker is a serverless application that runs on Cloudflare's edge network. In this setup, the Worker acts as the backend for your Telegram bot, receiving webhook updates from Telegram and processing commands, moderation logic, and AI responses.

This step focuses only on creating and deploying the Worker shell. Secrets and bindings will be configured in later steps.

2.1 Access the Cloudflare Dashboard

1. Open https://dash.cloudflare.com
2. Log in to your Cloudflare account
3. From the left-hand menu, select Workers & Pages

2.2 Create a New Worker

1. Click Create application
2. Select Worker (not Pages)
3. Click Create Worker
4. Enter a Worker name (example: telegram-group-bot)
5. Select the Hello World template

2.3 Replace the Default Code

1. Delete the Hello World code in the editor
2. Paste your Telegram bot Worker code
3. Ensure no secrets or tokens are hardcoded

2.4 Deploy the Worker

1. Click Save & Deploy
2. Wait for deployment to complete
3. Copy the public HTTPS Worker URL

2.5 Verify the Worker Is Live

1. Open the Worker URL in a browser
2. Confirm the Worker responds (any response is acceptable at this stage)
3. Note: Telegram is not connected yet

2.6 Common Mistakes to Avoid

- Forgetting to deploy after pasting code
- Leaving Hello World code in place
- Hardcoding bot tokens into the script
- Mismatched binding names

2.7 Final Checklist

✓ Worker created
✓ Code replaced
✓ Worker deployed
✓ HTTPS URL available

3. Configure Worker Bindings

Worker bindings allow your Cloudflare Worker to securely access secrets, KV storage, and Workers AI. Your Telegram bot will not function correctly until these bindings are properly configured.

This step connects the Telegram bot token, bad-word storage, and AI services to your Worker.

3.1 Open Worker Settings

1. Go to Cloudflare Dashboard
2. Navigate to Workers & Pages
3. Select your Worker
4. Click Settings
5. Open Variables & Bindings

3.2 Add Secret: TELEGRAM_BOT_TOKEN

1. Scroll to Secrets
2. Click Add
3. Variable name: TELEGRAM_BOT_TOKEN
4. Value: Paste your BotFather API token
5. Save

Notes:

• Secrets are encrypted
• Do not hardcode tokens in code

3.3 Create and Bind KV Namespace: BADLIST_KV

Create KV:

1. Go to Workers & Pages → KV
2. Click Create namespace
3. Name it (example: badlist)

Bind KV:

1. Return to Worker Settings → Variables & Bindings
2. Add KV namespace binding
3. Variable name: BADLIST_KV
4. Select your KV namespace

3.4 Add Workers AI Binding: AI

1. Scroll to AI bindings
2. Click Add binding
3. Variable name: AI
4. Save

Workers AI must be enabled on your account

3.5 Redeploy the Worker

1. Return to the Worker editor
2. Click Save & Deploy
3. Wait for deployment to finish

3.6 Verify Bindings

- No errors in Worker logs
- Bot token resolves
- KV reads succeed
- AI calls succeed

3.7 Common Mistakes

- Typo in binding names
- Forgetting to redeploy
- Token stored in code
- KV created but not bound

3.8 Final Checklist

✓ TELEGRAM_BOT_TOKEN secret added
✓ BADLIST_KV created and bound
✓ AI binding added
✓ Worker redeployed
✓ Logs clean

4. Add Bot to Telegram Group

After creating and deploying your bot, you must add it to your Telegram group and grant it the correct permissions. Without proper admin rights, the bot will not be able to moderate messages, manage users, or pin messages.

This step connects your bot to the group it will manage.

4.1 Add the Bot to Your Telegram Group

1. Open Telegram
2. Navigate to your target group
3. Open Group Info
4. Select Add members
5. Search for your bot by username (ending in 'bot')
6. Add the bot to the group

4.2 Promote the Bot to Admin

1. Open Group Info
2. Select Administrators
3. Tap Add Administrator
4. Choose your bot from the list
5. Review admin permissions

4.3 Enable Required Admin Permissions

Required permissions:

- Delete messages
- Ban users
- Restrict users
- Pin messages

Optional permissions:

- Invite users
- Manage video chats
- Manage topics

Do not enable Anonymous admin

4.4 Verify Bot Permissions

1. Send a test command (example: /about)
2. Confirm the bot responds
3. Test moderation actions if available

4.5 Notes on Supergroups

Most Telegram groups are supergroups by default.
Bots require supergroup status to moderate messages and users.
Telegram automatically upgrades groups when required.

4.6 Common Issues and Fixes

• Bot ignores commands → Not admin
• Cannot delete messages → Missing delete permission
• Mute/ban fails → Restrict/Ban not enabled
• Pin fails → Pin permission missing
• Bot silent → Webhook not active

4.7 Final Checklist

✓ Bot added to group
✓ Bot promoted to admin
✓ Delete messages enabled
✓ Ban users enabled
✓ Restrict users enabled
✓ Pin messages enabled
✓ Bot responds to commands

5. Register Webhook with Telegram

Telegram delivers updates (messages, joins, commands, etc.) to your bot using webhooks.
A webhook tells Telegram where to send these updates.

Your Cloudflare Worker acts as the webhook endpoint.

5.1 Deploy the Worker and get the HTTPS URL

1. Go to Cloudflare Dashboard → Workers & Pages
2. Open your Worker
3. Click Save & Deploy
4. Copy the public HTTPS URL provided

Requirements:

- Must be HTTPS
- Must be publicly reachable
- Must respond with HTTP 200 OK

5.2 Choose a secure webhook endpoint

Do not expose your webhook at the root path.
Use a secret path such as your bot token or a long random string.

Example:
https://your-worker.workers.dev/<SECRET_PATH>

5.3 Register the webhook with Telegram (setWebhook)

Option A: Browser
https://api.telegram.org/bot<YOUR_BOT_TOKEN>/setWebhook?url=<YOUR_WORKER_URL>

Option B: curl
curl -X POST
https://api.telegram.org/bot<YOUR_BOT_TOKEN>/setWebhook -d
url=<YOUR_WORKER_URL>

5.4 Verify webhook registration (getWebhookInfo)

https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getWebhookInfo

Check that the correct URL is shown and pending_update_count is 0

5.5 Common problems and fixes

• 403 Forbidden → Path mismatch
• Bot not responding → Missing admin permissions
• pending_update_count > 0 → Worker not responding

5.6 Check Worker logs

Cloudflare Dashboard → Workers & Pages → Logs
Send a Telegram message and confirm logs appear

5.7 Remove or reset webhook

https://api.telegram.org/bot<YOUR_BOT_TOKEN>/deleteWebhook

6. Enable Cloudflare Workers AI

Cloudflare Workers AI allows your Worker to run AI models directly at the edge. In this project, Workers AI is used to provide intelligent responses to user questions through the Telegram bot.

This step enables Workers AI on your Cloudflare account, binds it to your Worker, and ensures the Worker is redeployed so the AI functionality becomes active.

6.1 Enable Workers AI on Your Cloudflare Account

1. Log in to the Cloudflare Dashboard
2. From the left-hand menu, locate Workers & Pages
3. Select Workers AI
4. If prompted, click Enable Workers AI

Once enabled, your account can run supported AI models inside Workers.

6.2 Confirm AI Availability

After enabling Workers AI:

• Ensure your account plan supports Workers AI

• No API keys are required
• Billing (if applicable) is handled by Cloudflare automatically

6.3 Bind Workers AI to the Worker

1. Go to Workers & Pages
2. Select your Worker
3. Open Settings
4. Go to Variables & Bindings
5. Scroll to AI bindings
6. Click Add binding
7. Variable name: AI
8. Save changes

The binding name must be exactly 'AI' to match the Worker code.

6.4 Redeploy the Worker

1. Return to the Worker editor
2. Click Save & Deploy
3. Wait for deployment to complete

Bindings do not take effect until the Worker is redeployed.

6.5 Verify Workers AI Is Working

1. Open Worker Logs in the Cloudflare Dashboard
2. Send an AI-related command in Telegram (example: /ai)
3. Confirm no runtime errors appear
4. Confirm the bot responds with an AI-generated reply

6.6 Common Issues and Fixes

• AI binding missing → Add binding and redeploy
• Runtime error: env.AI undefined → Binding name mismatch
• No AI response → Workers AI not enabled on account
• Quota errors → Check Workers AI usage limits

6.7 Final Checklist

✓ Workers AI enabled on account
✓ AI binding added to Worker
✓ Worker redeployed
✓ AI responses working in Telegram

7. Create Cloudflare KV & Import BAD_WORDS JSON

Cloudflare KV is used to store persistent key-value data that your Worker can access at runtime. In this project, KV is used to store a JSON list of banned or sensitive words that the bot uses for moderation.

This step covers creating the KV namespace, binding it to the Worker, and importing the BAD_WORDS JSON data.

7.1 Create a Cloudflare KV Namespace

1. Log in to the Cloudflare Dashboard
2. Navigate to Workers & Pages
3. Select KV from the sidebar
4. Click Create namespace
5. Enter a name (example: badlist)
6. Create the namespace

7.2 Bind the KV Namespace to the Worker

1. Go to Workers & Pages
2. Select your Worker
3. Open Settings
4. Go to Variables & Bindings
5. Scroll to KV Namespace Bindings
6. Click Add binding
7. Variable name: BADLIST_KV
8. Select the KV namespace you created
9. Save changes

7.3 Prepare the BAD_WORDS JSON File

Create a JSON file containing an array of words or phrases.

Example format:

```
[
  "seed phrase",
  "private key",
  "airdrop bot",
  "free crypto"
]
```

Ensure the file is valid JSON and contains only strings.

7.4 Import BAD_WORDS into KV

1. Go to Workers & Pages → KV
2. Open your KV namespace
3. Click Add key-value pair
4. Key: BAD_WORDS
5. Value: Paste the JSON array contents
6. Save

The key name must be exactly BAD_WORDS.

7.5 Redeploy the Worker

1. Return to the Worker editor
2. Click Save & Deploy
3. Wait for deployment to complete

7.6 Verify KV Integration

1. Check Worker logs for KV-related errors
2. Send a test message containing a bad word
3. Confirm the bot detects or moderates the message

7.7 Common Issues and Fixes

- JSON parse errors → Invalid JSON format
- BAD_WORDS not found → Incorrect key name
- KV undefined → Binding name mismatch
- No moderation action → Logic disabled or permissions missing

7.8 Final Checklist

✓ KV namespace created
✓ KV bound as BADLIST_KV
✓ BAD_WORDS JSON imported
✓ Worker redeployed
✓ Moderation working

8. Add & Export BotFather Commands

BotFather commands define the list of commands users and admins see when typing '/' in Telegram. Properly configuring commands improves usability, discoverability, and reduces user confusion.

This step covers adding commands to BotFather and exporting the command list for backup or reuse.

8.1 Open BotFather Command Management

1. Open Telegram
2. Search for @BotFather
3. Open the chat and send /start
4. Send /mybots
5. Select your bot
6. Choose Edit Bot → Edit Commands

8.2 Add Commands to BotFather

1. BotFather will prompt you to enter commands
2. Enter one command per line using the format:
   command - description

Example:
rules - Show group rules
about - About the project
ai - Ask the AI assistant
mute - Admin: mute a user

8.3 Example Full Command List

Public commands:
rules - Show group rules
links - Official links
about - About the project
faq - Frequently asked questions
tokenomics - Token details
ai - Ask the AI assistant

Moderator/Admin commands:
addmod - Add moderator (reply)
removemod - Remove moderator (reply)
mods - List moderators
ismod - Check if user is moderator
mute - Mute user (reply)
unmute - Unmute user (reply)
warn - Warn user
kick - Kick user
ban - Ban user
purge - Delete recent messages
away - Away mode control

8.4 Save and Apply Commands

1. After pasting the command list, send it to BotFather
2. BotFather will confirm commands were updated
3. Commands become active immediately

8.5 Verify Commands in Telegram

1. Open your Telegram group
2. Type '/'
3. Confirm commands appear in the suggestion list

8.6 Export Commands for Backup

Best practices for exporting commands:

- Save the command list in a text file
- Store it in version control or documentation
- Keep a copy in your SOP or admin guide

BotFather does not provide an automatic export feature.

8.7 Common Issues and Fixes

- Commands not visible → Bot not added to group
- Commands missing → BotFather list incomplete
- Wrong descriptions → Re-run Edit Commands

8.8 Final Checklist

✓ Commands added in BotFather
✓ Public and admin commands documented
✓ Commands visible in Telegram
✓ Command list backed up

9. Using and Maintaining the Command List (Admin Workflow)

This section explains how administrators and moderators should use, maintain, and update the Telegram bot command list over time. A clear workflow ensures consistency, prevents misuse, and keeps commands aligned with bot functionality.

9.1 Understanding Command Types

Commands generally fall into two categories:

Public commands:

• Available to all group members
• Used for information and help

Admin/Moderator commands:
• Restricted to admins or mods
• Used for moderation and group management

9.2 Daily Admin Usage Workflow

Recommended daily workflow:
1. Use public commands to answer common questions
2. Use moderation commands via reply to target users
3. Monitor bot responses for errors or delays
4. Review warnings, mutes, and bans issued by the bot

9.3 Adding New Commands

When new commands are added to the Worker code:
1. Update the Worker code and deploy
2. Open @BotFather
3. Use /mybots → Edit Bot → Edit Commands
4. Add the new command and description
5. Save and verify in Telegram

9.4 Updating Command Descriptions

If a command's behavior changes:
1. Update the description in BotFather
2. Keep descriptions short and clear
3. Avoid exposing internal or admin-only logic

9.5 Removing Deprecated Commands

When a command is no longer supported:
1. Remove it from the Worker code
2. Remove it from BotFather command list
3. Notify admins of the change
4. Update documentation

9.6 Command Documentation Best Practices

• Maintain a master command list document
• Store it in version control or shared storage
• Include command purpose and permission level
• Update documentation with every release

9.7 Permission and Safety Guidelines

• Never grant admin commands to untrusted users
• Review moderator list regularly
• Log or audit moderation actions if possible
• Revoke access immediately if abuse is detected

9.8 Training New Admins and Mods

Recommended onboarding steps:

1. Provide this SOP document
2. Explain reply-based moderation commands
3. Clarify escalation rules (warn → mute → ban)
4. Test commands in a staging group if available

9.9 Common Mistakes and Fixes

• Commands not visible → BotFather list outdated
• Admin misuse → Clarify permissions and rules
• Bot rejects command → Permission or syntax error

9.10 Final Checklist (Admin Workflow)

✓ Command list documented
✓ BotFather commands up to date
✓ Admins trained
✓ Deprecated commands removed
✓ Workflow reviewed regularly

10. Exported Command List

rules - Show rules
links - Official links
about - About project

faq - Frequently asked questions
tokenomics - Token details
ai - Ask AI
addmod - Add moderator
removemod - Remove moderator
mods - List moderators
ismod - Check moderator
mute - Mute user
unmute - Unmute user
warn - Warn user
kick - Kick user
ban - Ban user
purge - Delete messages
away - Away mode