Claude Code Channels

Complete VPS Setup Guide

Turn Your AI Into an Always-On Assistant

You Can Text From Your Phone

By LLM Match Maker | llmmatchmaker.com

Tested on: Claude Code v2.1.84, Telegram plugin v0.0.4, Ubuntu 24.04 LTS, Node.js 22
Last updated March 2026

What You'll Build

By the end of this guide, you'll have:

- Claude Code running 24/7 on a VPS
- A Telegram bot that responds to your messages with full AI capabilities
- Auto-restart on reboot (no manual intervention needed)

Total cost: $6.99/mo (VPS) + your existing Claude subscription.

A note before you start: This guide is a working proof of concept based on our actual setup. It will get you a functional Telegram bot connected to Claude Code on a VPS. However, your environment may differ. Software versions change, Anthropic updates their plugin system, and VPS configurations vary. Treat this as a foundation to build on, not a guaranteed copy-paste solution. Test everything in your own environment and adapt as needed.

Prerequisites

- A Claude subscription (Pro, Max, Team, or Enterprise)
- A Telegram account
- Basic comfort with SSH and terminal commands

Step 1: Get a VPS

You need a Linux server that runs 24/7. Any Ubuntu VPS works.

Recommended: Hostinger KVM 1 ($6.99/mo) with 4GB RAM is more than enough for Claude Code Channels.

If you also want to run n8n automation workflows, grab a KVM 2 ($9.99/mo) with 8GB RAM.

After purchasing, note your server IP address and set up SSH key authentication.

Step 2: Install Claude Code on the VPS

SSH into your server:

```
ssh your-username@your-server-ip
```

Install Node.js 22:

```
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt install -y nodejs
```

Install Claude Code:

```
sudo npm install -g @anthropic-ai/claude-code@latest
```

Verify:

```
claude --version
```

Install Bun (required for the Telegram plugin):

```
curl -fsSL https://bun.sh/install | bash
```

Important: If you're running as a non-root user, make sure Bun is accessible system-wide:

```
sudo cp ~/.bun/bin/bun /usr/local/bin/bun
sudo chmod 755 /usr/local/bin/bun
```

> **Gotcha #1:** If you only symlink Bun to root's home directory, other users can't follow the symlink. Always copy the actual binary.

Step 3: Authenticate Claude Code

On the VPS (not your local machine), run:

```
claude setup-token
```

This opens a browser authentication flow. Complete it and save the OAuth token it generates. Then set it in your shell:

```
export CLAUDE_CODE_OAUTH_TOKEN=your-token-here
echo 'export CLAUDE_CODE_OAUTH_TOKEN=your-token-here' >> ~/.bashrc
```

> **Gotcha #2:** You must run claude setup-token ON the VPS. Running it locally generates a token for your local machine, not the server.

Step 4: Create a Telegram Bot

1. Open Telegram and search for **@BotFather**
2. Send `/newbot`
3. Choose a display name (e.g., "My AI Assistant")
4. Choose a username ending in `bot` (e.g., `myai_assistant_bot`)
5. Copy the bot token BotFather gives you (format: `1234567890:AAH...`)

Save this token. You'll need it next.

Step 5: Install the Telegram Plugin

This must be done from INSIDE a Claude Code session, not from the command line.

Start Claude Code:

```
claude
```

Once loaded, type:

```
/plugin install telegram@claude-plugins-official
```

Select "Install for you (user scope)" and confirm. Exit Claude Code:

```
/exit
```

> **Gotcha #3:** Running claude plugin install telegram@claude-plugins-official from the shell fails with "not found in marketplace." The /plugin install command inside Claude Code uses a different discovery mechanism.

Step 6: Configure the Bot Token

Start Claude Code with the channels flag:

```
claude --channels plugin:telegram@claude-plugins-official
```

Once loaded, type:

```
/telegram:configure YOUR_BOT_TOKEN_HERE
```

This saves the token to `~/.claude/channels/telegram/.env`.

Step 7: Pair Your Telegram Account

With Claude Code still running (from Step 6):

1. Open Telegram on your phone
2. Find your bot and send it any message (e.g., “hello”)
3. The bot replies with a 6-character pairing code
4. In Claude Code, type: `/telegram:access pair THE_CODE`
5. Lock it down: `/telegram:access policy allowlist`

Now only your Telegram account can message the bot. Exit Claude Code:

```
/exit
```

Step 8: Set Permissions (Prevent Blocking)

Claude Code asks for permission before using tools. In a background session, these prompts freeze the bot. Set broad permissions.

Create `~/.claude/settings.local.json`:

```
{
  "permissions": {
    "allow": ["*"]
  }
}
```

Also create the project-level settings file with the same content:

```
mkdir -p ~/.claude/projects/-home-yourusername
cp ~/.claude/settings.local.json ~/.claude/projects/-home-
yourusername/settings.local.json
```

> **Gotcha #4:** The * wildcard in settings doesn’t cover MCP tool permissions in all versions. You may still get prompts on first use. The auto-approval watcher (Step 10) handles this.

Step 9: Make It Permanent (systemd Service)

Create a systemd service so the bot starts on boot:

```
sudo nano /etc/systemd/system/claude-channel.service
```

Paste the following (replace `yourusername` and `your-oauth-token`):

```
[Unit]
Description=Claude Code Telegram Channel
After=network.target

[Service]
Type=forking
User=yourusername
Group=yourusername
WorkingDirectory=/home/yourusername
Environment=HOME=/home/yourusername
Environment=CLAUDE_CODE_OAUTH_TOKEN=your-oauth-token
```

```
Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
ExecStart=/bin/bash -c "cd /home/yourusername && /usr/bin/tmux new-session -d -s channel '/usr/bin/claude --channels plugin:telegram@claude-plugins-official' && sleep 2 && nohup /home/yourusername/channel-watcher.sh > /tmp/watcher.log 2>&1 &"
ExecStop=/bin/bash -c "/usr/bin/tmux kill-session -t channel 2>/dev/null; pkill -f channel-watcher.sh 2>/dev/null"
Restart=on-failure
RestartSec=30
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Enable and start:

```
sudo systemctl daemon-reload
sudo systemctl enable claude-channel
sudo systemctl start claude-channel
```

> **Gotcha #5:** Claude Code requires an interactive terminal (TTY). That's why we use tmux inside systemd, not a direct ExecStart. Without tmux, Claude Code exits immediately.

Step 10: Auto-Approve Permission Prompts

There are two layers of permission prompts to handle: **terminal prompts** (Claude Code asking in the tmux session) and **Telegram prompts** (the Channels plugin forwarding permission requests as Allow/Deny buttons in your chat). You need to fix both.

Part A: Terminal Watcher Script

Create the watcher script that monitors the tmux session and auto-approves any terminal permission prompts:

```
nano ~/channel-watcher.sh
```

Paste the following script:

```
#!/bin/bash
SESSION="channel:0"

while true; do
    OUTPUT=$(tmux capture-pane -t "$SESSION" -p 2>/dev/null)
    if [ $? -ne 0 ]; then
        sleep 5
        continue
    fi
    if echo "$OUTPUT" | grep -q "Do you want to"; then
        if echo "$OUTPUT" | grep -q "don't ask again"; then
            tmux send-keys -t "$SESSION" Down Enter
        else
            tmux send-keys -t "$SESSION" Enter
        fi
        sleep 1
        continue
```

```
    fi
    if echo "$OUTPUT" | grep -q "Yes, I trust this folder"; then
        tmux send-keys -t "$SESSION" Enter
        sleep 1
        continue
    fi
    if echo "$OUTPUT" | grep -q "Yes, I accept"; then
        tmux send-keys -t "$SESSION" Down Enter
        sleep 1
        continue
    fi
    if echo "$OUTPUT" | grep -q "Use skill"; then
        if echo "$OUTPUT" | grep -q "don't ask again"; then
            tmux send-keys -t "$SESSION" Down Enter
        else
            tmux send-keys -t "$SESSION" Enter
        fi
        sleep 1
        continue
    fi
    sleep 1
done
```

Make it executable:

```
chmod +x ~/channel-watcher.sh
```

> **Gotcha #6:** The permission prompt text changes between Claude Code versions. The updated watcher uses "Do you want to" to catch all variants. The sleep is reduced from 3s to 1s for faster approval.

Part B: Patch the Telegram Plugin

Even with the terminal watcher working, the Channels plugin sends permission requests as Allow/Deny buttons in your Telegram chat. To make the bot fully autonomous, patch the plugin to auto-approve all permissions.

Back up and locate the plugin source:

```
PLUGIN_FILE=~/.claude/plugins/marketplaces/claude-plugins-
official/external_plugins/telegram/server.ts
cp "$PLUGIN_FILE" "${PLUGIN_FILE}.bak"
```

Create and run the patch script (see guide source for full Python patch). After patching, restart:

```
sudo systemctl restart claude-channel
```

> **Gotcha #7:** This patch removes the Allow/Deny buttons from Telegram entirely. All tool calls execute immediately without asking. This is the right choice for a private bot on an allowlisted account, but do not use this on a bot that accepts messages from untrusted users.

Step 11: Add a CLAUDE.md (Optional but Recommended)

Create `~/CLAUDE.md` to give your bot personality and default behaviors:

```
nano ~/CLAUDE.md
```

Example content:

```
# My AI Assistant

You are a helpful AI assistant running on a VPS via Telegram.

## Default Behaviors
- Keep responses concise for Telegram
- When someone sends a file, analyze it and provide a summary
- When asked about server status, run the appropriate commands
```

Step 12: Verify Everything Works

Check the service:

```
sudo systemctl status claude-channel
```

Attach to the tmux session to see what's happening:

```
tmux attach -t channel
```

Detach without killing it: `Ctrl+B` then `D`

Send a message to your bot on Telegram. It should respond within seconds.

Management Commands

Command	What it does
`sudo systemctl status claude-channel`	Check if bot is running
`sudo systemctl restart claude-channel`	Restart the bot
`sudo systemctl stop claude-channel`	Stop the bot
`journalctl -u claude-channel`	View service logs
`tmux attach -t channel`	See the live Claude Code session
`Ctrl+B then D`	Detach from tmux (bot keeps running)

Troubleshooting

Bot doesn't respond:

1. Check service: `sudo systemctl status claude-channel`
2. Check MCP server: Attach to tmux, type `/mcp`
3. Check watcher log: `cat /tmp/watcher.log`

"1 MCP server failed":

- Bun is not in PATH. Run: `sudo cp ~/.bun/bin/bun /usr/local/bin/bun`

Bot freezes on a prompt:

- Watcher script may have stopped. Check: `ps aux | grep watcher`
- Restart: `nohup ~/channel-watcher.sh > /tmp/watcher.log 2>&1 &`

Allow/Deny buttons showing in Telegram:

- The Telegram plugin patch (Step 10 Part B) was not applied or was overwritten by a plugin update
- Re-apply the patch and restart: `sudo systemctl restart claude-channel`

OAuth token expired:

- Tokens last 1 year. Regenerate: `claude setup-token` (on the VPS)
- Update the token in the systemd service file, then run: `sudo systemctl daemon-reload && sudo systemctl restart claude-channel`

What's Next?

Now that you have an always-on AI assistant, explore what else it can do:

- **Run server commands** from your phone
- **Draft content** while commuting
- **Search the web** and get summarized results
- **Manage files** on your server
- **Analyze documents** by sending PDFs via Telegram
- **Build custom skills** to automate repetitive tasks

For pre-built automation workflows that extend this setup with n8n, check out our Pro Tracks at llmmatchmaker.com/playbooks.

Built and documented by LLM Match Maker. Questions? Visit llmmatchmaker.com