Step-by-Step Guide: Build a GitHub App that Auto-Comments on Issues

This guide recreates the working "Hello World" GitHub App that automatically comments on newly opened issues.

Prerequisites: Python 3.10+, GitHub account, 30 minutes.

Step 1: Register GitHub App (5 min)

1. Go to https://github.com/settings/apps → **New GitHub App**
2. Fill exactly like this:

GitHub App name: simple-issue-bot
Homepage URL: https://github.com/YOUR_USERNAME/simple-issue-bot
Webhook URL: https://example.com/webhook (placeholder for now)
Webhook secret: mysecret123 (copy this!)
Active: ☑ checked
Permissions → Repository permissions:
• Issues: Read & write
• (All others: No access)
Subscribe to events:
• ☑ Issues
Where can be installed: Only on this account

3. **Create GitHub App**
4. **Generate private key** → Download `simple-issue-bot.2026-01-19.private-key.pem`
5. **Note App ID** (top of page, e.g. `123456`)

Step 2: Setup Python Project (3 min)

mkdir simple-github-app
cd simple-github-app
python3 -m venv .venv
source .venv/bin/activate # macOS/Linux

.venv\Scripts\activate # Windows

pip install flask pyjwt cryptography requests

Rename downloaded PEM to `private-key.pem` and put in folder.

Step 3: Create the Code Files

app.py

```
import os
import hmac
import hashlib
import time
from flask import Flask, request, abort
import requests
from jwt_utils import make_jwt

app = Flask(name)
```

Load from environment

```
GITHUB_APP_ID = os.environ["GITHUB_APP_ID"]
GITHUB_WEBHOOK_SECRET = os.environ["GITHUB_WEBHOOK_SECRET"].encode()
GITHUB_INSTALLATION_ID = os.environ["GITHUB_INSTALLATION_ID"]

with open("private-key.pem", "r", encoding="utf-8") as f:
PRIVATE_KEY = f.read()

def get_installation_token():
jwt_token = make_jwt(GITHUB_APP_ID, PRIVATE_KEY)
url = f"https://api.github.com/app/installations/{GITHUB_INSTALLATION_ID}/access_tokens"
headers = {
"Authorization": f"Bearer {jwt_token}",
"Accept": "application/vnd.github+json"
}
resp = requests.post(url, headers=headers, timeout=10)
resp.raise_for_status()
return resp.json()["token"]

def verify_signature(payload_body: bytes, signature_header: str | None) -> bool:
if not signature_header:
return False
mac = hmac.new(GITHUB_WEBHOOK_SECRET, msg=payload_body,
digestmod=hashlib.sha256)
expected = "sha256=" + mac.hexdigest()
return hmac.compare_digest(expected, signature_header)

def comment_on_issue(owner: str, repo: str, issue_number: int) -> None:
token = get_installation_token()
url = f"https://api.github.com/repos/{owner}/{repo}/issues/{issue_number}/comments"
headers = {
"Authorization": f"Bearer {token}",
"Accept": "application/vnd.github+json"
}
body = {"body": "Hello from my GitHub App! □"}
```

```
resp = requests.post(url, headers=headers, json=body, timeout=10)
resp.raise_for_status()
```

```
@app.route("/webhook", methods=["POST"])
def webhook():
payload = request.data
signature = request.headers.get("X-Hub-Signature-256")
if not verify_signature(payload, signature):
abort(401)
```

```
event = request.headers.get("X-GitHub-Event")
data = request.get_json()

if event == "issues" and data.get("action") == "opened":
    owner = data["repository"]["owner"]["login"]
    repo = data["repository"]["name"]
    issue_number = data["issue"]["number"]
    comment_on_issue(owner, repo, issue_number)

return "", 204
```

if **name** == "**main**":
app.run(host="0.0.0.0", port=3000)

jwt_utils.py

```
import jwt
import time
```

```
def make_jwt(app_id: str, private_key: str) -> str:
"""
Create a short-lived JWT for the GitHub App.
```

```
GitHub requires:
- 'iss' = app ID
- 'iat' = issued at (<= 60s in the past)
- 'exp' = expiration (<= 10 minutes in the future)
"""
now = int(time.time())
payload = {
    "iat": now - 60,
    "exp": now + 600,
    "iss": app_id,
}
encoded = jwt.encode(payload, private_key, algorithm="RS256")
if isinstance(encoded, bytes):
    encoded = encoded.decode("utf-8")
return encoded
```

Step 4: Set Environment Variables (1 min)

export GITHUB_APP_ID=123456 # From Step 1 #5
export GITHUB_INSTALLATION_ID=????? # From Step 6 #2
export GITHUB_WEBHOOK_SECRET=mysecret123 # From Step 1 #2

Step 5: Run Locally (2 min)

python app.py

Expected output:

- Running on all addresses (0.0.0.0)
- Running on http://127.0.0.1:3000

Step 6: Expose Publicly with ngrok (3 min)

In a new terminal (keep Flask running):

1. Install ngrok: https://ngrok.com/download
2. Create a free account: https://dashboard.ngrok.com/signup
3. Get your authtoken: https://dashboard.ngrok.com/get-started/your-authtoken
4. Configure locally (one-time):

ngrok config add-authtoken YOUR_AUTHTOKEN_HERE

5. Start the tunnel:

ngrok http 3000

Copy the URL:
Forwarding https://abc123.ngrok-free.app -> http://localhost:3000

Your webhook URL: `https://abc123.ngrok-free.app/webhook`

Step 7: Configure Webhook (1 min)

1. Go to your GitHub App settings
2. Update **Webhook URL**: `https://abc123.ngrok-free.app/webhook`
3. Confirm **Webhook secret**: mysecret123 (matches env var)
4. **Save changes**

Step 8: Install on Test Repo & Get Installation ID (2 min)

1. From GitHub App page → **Install App**
2. Select **Only select repositories** → pick a test repo → **Install**
3. Go to https://github.com/settings/installations
4. Click **Configure** next to your app
5. URL shows `.../installations/987654321`
6. Copy that number (e.g., `987654321`)
7. Update env variable:

```
export GITHUB_INSTALLATION_ID=987654321
```

8. **Restart Flask**: Ctrl+C then `python app.py`

Step 9: Test! (30 seconds)

1. Go to the test repo where you installed the app
2. Create a **new issue** (any title or description)
3. **Expected result**: Bot automatically comments "Hello from my GitHub App! ▯"
4. ngrok dashboard (http://127.0.0.1:4040) shows `POST /webhook 204`

Success! ▯

How It Works (Architecture)

The flow when an issue is opened:
1. **GitHub sends webhook** → POST `https://abc123.ngrok.io/webhook` with issue payload
2. **Flask receives** → verifies HMAC signature (prevents spoofing)
3. **App generates JWT** → signs with private key (proves app identity)
4. **JWT** → **installation token** → POST to `/app/installations/{id}/access_tokens`
5. **Uses installation token** → comments on issue via REST API
6. **Returns 204** → GitHub marks delivery successful
Each installation token is short-lived (1 hour) and scoped to the app's permitted repositories.

Troubleshooting

Problem	**Solution**
404 Not Found /installations/XXXX	Wrong Installation ID (check Step 8 #5)
401 signature failed	Webhook secret mismatch (Step 1 #2 vs Step 4)
500 JWT error	Wrong App ID or corrupted private-key.pem
No comment appears	Check ngrok URL is active, ngrok dashboard for failed requests
Flask won't start	Missing env vars or private-key.pem not in folder
ngrok says 404	GitHub App webhook URL not updated (Step 7)

Table 1: Common errors and fixes

Debug steps:

- Check Flask console for stack trace (most detailed)
- Check GitHub App → **Advanced** → **Webhook deliveries** (see status + response)
- Check ngrok dashboard: http://127.0.0.1:4040 (see request/response)

Security Best Practices

- **Private key**: Never commit to git; store in secrets manager
- **Webhook secret**: Should be random, 20+ characters
- **Least privilege**: Only "Issues: Read & write", no other permissions
- **Installation tokens**: Expire in 1 hour (GitHub enforces)
- **Signature verification**: Always verify HMAC before processing

---

Optional: Upgrade to PyGithub (Bonus)

For cleaner code, use PyGithub library instead of raw requests:

```
pip install PyGithub
```

Replace `comment_on_issue()` function:

```
from github import Github
```

```
def comment_on_issue(owner: str, repo: str, issue_number: int) -> None:
token = get_installation_token()
g = Github(token)
repository = g.get_repo(f"{owner}/{repo}")
issue = repository.get_issue(issue_number)
issue.create_comment("Hello from PyGithub! ▯")
```

Next Steps

- **Deploy**: Use Heroku/Vercel so ngrok isn't needed (store secrets as env vars)
- **Extend**: Add label logic, PR checks, auto-close stale issues
- **Interview prep**: Practice explaining the flow above; this is a complete working example

References

[1] GitHub Apps Overview - https://docs.github.com/en/apps/overview

[2] Registering a GitHub App - https://docs.github.com/en/apps/creating-github-apps/registering-a-github-app/registering-a-github-app

[3] Using webhooks with GitHub Apps - https://docs.github.com/en/apps/creating-github-apps/registering-a-github-app/using-webhooks-with-github-apps

[4] Authenticating as a GitHub App Installation - https://docs.github.com/en/apps/creating-github-apps/authenticating-with-a-github-app/authenticating-as-a-github-app-installation

[5] Webhook events and payloads - https://docs.github.com/en/webhooks/webhook-events-and-payloads

[6] PyGithub Documentation - https://pygithub.readthedocs.io/en/latest/

[7] ngrok documentation - https://ngrok.com/docs