# 🧾 Customer Repo Triage Automation Guide

This guide helps you:

- Fork a customer repo you can’t modify

- Sync code and issues regularly

- Use GitHub Copilot to triage and suggest fixes

- Automate everything with GitHub Actions

---

## 📌 Overview

| Task | Tool |

|------|------|

| Fork customer repo | GitHub |

| Sync code | GitHub Actions |

| Sync issues | Python + GitHub API |

| Triage issues | GitHub Copilot |

| Automate everything | GitHub Actions |

---

## ✅ Step 1: Fork the Customer Repository

1. Go to the customer’s GitHub repository.

2. Click \*\*Fork\*\* and choose your GitHub Enterprise (GHE) account or organization.

3. This creates a forked repo under your control (e.g., `your-org/customer-repo-fork`).

---

## 🔐 Step 2: Create a GitHub Personal Access Token (PAT)

1. Go to GitHub Developer Settings.

2. Click \*\*"Generate new token (classic)"\*\*.

3. Select scopes:

- `repo` (full control of private repositories)

- `workflow` (for GitHub Actions)

4. Copy and save the token securely.

---

## 🧰 Step 3: Set Up Python Environment

### a. Install Python

1. Download the latest Python 3.11+ from python.org.

2. Run the installer:

- ✅ Check \*\*“Add Python to PATH”\*\*

- Click \*\*Customize installation\*\*

- Ensure \*\*pip\*\* is selected

- Click \*\*Install\*\*

### b. Verify Installation

Open Command Prompt and run:

```bash

python --version

pip --version

You should see something like:

Python 3.11.6 pip 23.x.x

### c. Install Required Packages

pip install PyGithub requests

## 🧠 Step 4: Create the Issue Sync Script

Create a file named sync\_issues.py and paste the following (check the repo names, replacing <yourorg/yourrepo> :

*import os*

*from github import Github*

*import requests*

*# === CONFIGURATION ===*

*GITHUB\_TOKEN = os.getenv("GITHUB\_TOKEN") # Or hardcode for testing*

*UPSTREAM\_REPO = "ssc-spc-ccoe-cei/azure-guardrails-solution-accelerator"*

*FORK\_REPO = "<yourorg/yourrepo>”*

*LABEL\_PREFIX = "[Upstream]"*

*# === INIT ===*

*gh = Github(GITHUB\_TOKEN)*

*upstream = gh.get\_repo(UPSTREAM\_REPO)*

*fork = gh.get\_repo(FORK\_REPO)*

*# === FETCH ISSUES FROM UPSTREAM ===*

*upstream\_issues = upstream.get\_issues(state="open")*

*# === SYNC TO FORK ===*

*for issue in upstream\_issues:*

*# Check if issue already exists in fork (by title or custom label)*

*existing = fork.get\_issues(state="all", labels=[LABEL\_PREFIX])*

*if any(i.title == issue.title for i in existing):*

*continue # Skip if already synced*

*# Create issue in fork*

*# Check if triage label exists and prepare labels list*

*available\_labels = [l.name for l in fork.get\_labels()]*

*labels\_to\_apply = []*

*if "triage" in available\_labels:*

*labels\_to\_apply.append("triage")*

*fork.create\_issue(*

*title=f"{LABEL\_PREFIX} {issue.title}",*

*body=f"\*\*Original Issue:\*\* {issue.html\_url}\n\n{issue.body}",*

*labels=labels\_to\_apply*

*)*

*print("Issues synced successfully.")*

## ⚙️ Step 5: OPTIONAL: Automate with GitHub Actions

### a. Create Workflow File

In your forked repo, create:

.github/workflows/sync\_issues.yml

### b. Paste This Workflow:

name: Sync Issues from Upstream

on:

schedule:

- cron: '0 12 \* \* \*' # Daily at 12:00 UTC

workflow\_dispatch:

jobs:

sync:

runs-on: ubuntu-latest

steps:

- name: Checkout

uses: actions/checkout@v3

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: '3.x'

- name: Install dependencies

run: pip install PyGithub requests

- name: Run sync script

env:

GITHUB\_TOKEN: ${{ secrets.PERSONAL\_ACCESS\_TOKEN }}

run: |

curl -O https://raw.githubusercontent.com/your-org/your-repo/main/sync\_issues.py

python sync\_issues.py

## 🔐 Step 6: OPTIONAL: Add Your PAT to GitHub Secrets

1. Go to your forked repo on GitHub.
2. Click **Settings > Secrets and variables > Actions**.
3. Click **New repository secret**.
4. Name it: PERSONAL\_ACCESS\_TOKEN
5. Paste your GitHub PAT.

## 🧠 Step 7: Triage with Copilot

Now that issues are synced:

* Use **GitHub Copilot Chat** in VS Code to analyze issues.
* Create branches and draft PRs in your fork.
* Share findings with the customer via reports or PRs.