

<GIT_SITUATION>
Comprehensive Analysis and Strategy for GitHub Push Protection Failure 🚫

This document compiles the history, analyzes the persistent error, and proposes ranked strategies for an experienced GitHub/Git expert to quickly resolve the push failure due to GitHub's Secret Scanning.

1. Fundamental Problem and Error Persistence

A. The Core Violation (The Ghost Secret)

The repository push is blocked by GitHub Push Protection because a specific Anthropic API Key was introduced into the repository's history and is detected in five specific configuration files.

- Violating Commit Hash: The error consistently names the original offending commit hash as e6b58a1962c22656566f92276b8dbf9f450fcf91 (which is the current rewritten version of the original f52aecd commit).
- Violating Files (All contain the secret on line 3, 4, or 10):

```
1. configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy-info.json
2. configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy.json
3. configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-long.json
4. configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json
5. configuration/dot-claude_snapshot/macos_20251112-0912est/settings.json
```

B. The Persistence Challenge

The error persists because the core cleanup step—manually replacing the token with a placeholder inside the files at the exact moment of the first edit stop—has failed or been undone at every rebase attempt. The push failure confirms that the actual secret token (e.g., sk-ant-api-01...) remains on the five specific lines of commit e6b58a19...

2. History of Fix Attempts and Key Learnings

Attempt Description Status/Error Key Learning Initial Attempts (5-15 pushes) Remote Rejected Simply committing a new file state does not rewrite history. Nested Git Issue git rm --cached failed, directory conflict ensued. Directory structural conflicts must be resolved by deleting the internal .git folder and force-deleting Git's temporary conflict directories (rm -rf). Interactive Rebase #1 Rebase completed without stopping (pick not changed to edit). Crucial: The edit command in the rebase script must be explicitly set to pause for manual file amendment. Interactive Rebase #2 & #3 Successful rebase continuation, but final push failed. Core Error: Although the rebase mechanism works now, the actual file content change (replacing the secret with a non-matching placeholder) was missed or reverted in the initial git commit --amend stop. Final State (Current) Successful rebase completion, but push blocked again on the same hash and same files. The history is now linear and clean of structural conflicts, but the content of the first amended commit (e6b58a19...) is still dirty.

3. Current Repository State and Problem Status

- Current Branch: main (ahead of origin/main).
- History State: The local history has been fully rebased and cleaned of structural issues, moving all commits onto f0ab488.
- Problem Status: The issue is a pure content-level secret leak. The history rewrite procedure is correct, but the manual intervention step (editing the file contents) needs to be performed with certainty.

4. Ranked Strategies for Expert Fix (Recommended Path)

💡 Strategy 1: Targeted Interactive Rebase (Most Surgical)

This confirms the local history is correct and forces the expert to execute the one missing step: file content amendment.

1. Start Rebase: Rebase again, targeting the parent of the currently failing commit.

Bash

```
&nbsp;&nbsp;&nbsp;git rebase -i f0ab488 &nbsp;&nbsp;&nbsp;
```

2. Edit Script: In the editor, explicitly change pick to edit for the first commit (e6b58a19... or its current hash).

3. Clean Files (The CRITICAL step) 🚫: At the edit stop, the expert MUST manually open the five files listed in the GitHub error and replace the actual Anthropic API Key (which is present) with a safe, non-matching placeholder string like "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER".

4. Amend and Continue:

Bash

```
&nbsp;&nbsp;&nbsp;git add ... &nbsp;&nbsp;&nbsp;git commit --amend --no-edit
&nbsp;&nbsp;&nbsp;git rebase --continue &nbsp;&nbsp;&nbsp;
```

5. Force Push:

Bash

```
&nbsp;&nbsp;&nbsp;git push --force-with-lease &nbsp;&nbsp;&nbsp;
```

⚠ Strategy 2: BFG Repo-Cleaner (Automated Global Cleanup)

If Strategy 1 fails or if the secret is suspected to be in other commits/files, the BFG Repo-Cleaner is the standard tool for deleting secrets from Git history efficiently.

1. Download BFG (or use the packaged JAR file).
2. Clone Mirror: Clone a mirror of the repository.
Bash

```
&nbsp;&nbsp;&nbsp;git clone --mirror <repo_url> &nbsp;&nbsp;&nbsp;
```

3. Run BFG Command: Use BFG to replace the exact secret string with a placeholder across all branches.
Bash

```
&nbsp;&nbsp;&nbsp;java -jar bfg.jar --replace-text
<file_containing_secret_string> --all &nbsp;&nbsp;&nbsp;
```

4. Cleanup & Force Push:
Bash

```
&nbsp;&nbsp;&nbsp;cd <repo_name>.git &nbsp;&nbsp;&nbsp;git reflog expire --
expire=now --all && git gc --prune=now --aggressive &nbsp;&nbsp;&nbsp;git push
&nbsp;&nbsp;&nbsp;
```

⚠ Strategy 3: GitHub Bypass (Last Resort/Quick Fix)

The least clean solution, but the fastest, is to use the URL provided by GitHub to unblock the push for this specific secret.

1. Navigate: Go to the URL provided in the terminal output: <https://github.com/ion-chun/clause-code-configs/security/secret-scanning/unblock-secret/35PVnYwS8h7MURZI4sVhXE69je>
2. Bypass: Choose the option to "Bypass protection" (e.g., "This secret is used for testing").
3. Push: Rerun the git push --force-with-lease command.

This is not a true fix, as the secret remains in the history, but it allows the push to succeed immediately.

</GIT_SITUATION>

<INSTRUCTIONS>

Think hard to carefully analyze this <GIT_SITUATION> that another AI LLM cannot seem to resolve/fix, ask me any clarifying question if you need more info/details, and then give a step by step way to solve this git issue

</INSTRUCTIONS>

💡 Thought for 8s

After carefully reviewing the provided <GIT_SITUATION> document, I see a clear pattern in why previous attempts (likely guided by another AI) have failed to resolve this: the core issue isn't the rebase process itself—it's the incomplete or ineffective manual intervention during the "edit" step in the interactive rebase. Specifically, the secret token hasn't been fully replaced in the files at that exact historical point, or the changes weren't committed properly before continuing the rebase. This leads to the rewritten commit (e6b58a19...) still containing the detectable secret, as GitHub's scanning checks the entire history, not just the latest state. Other factors like nested Git conflicts were red herrings that got resolved, but they distracted from the content-level fix. Automated tools like BFG weren't tried, but they're overkill if we can surgically edit via rebase.

The document's Strategy 1 is the right path (most precise and least disruptive), but it needs more explicit safeguards to ensure the edit step succeeds without reversion. I'll expand it into a foolproof step-by-step guide below, with verification checks at each stage to catch any slips. This assumes you're on macOS (based on the file paths) and have Git installed/configured properly. If you're on a different OS or using a Git GUI, let me know for adjustments.

Before proceeding: a couple clarifying questions to make this airtight—

1. Can you run `git log --oneline --graph -n 10` and share the output? This will confirm the current commit structure (e.g., if e6b58a19... is indeed the first commit after f0ab488).
2. Is the secret token exactly "sk-ant-api-01..." (as hinted), or a variant? Knowing the precise string helps in search/replace commands.
3. Have you already tried the GitHub bypass URL, or do you want to avoid it to truly clean the history?

If you provide answers, I can refine this further. Otherwise, proceed with the steps below on a backup of your repo (e.g., `git clone` to a new folder) to avoid risks.

Step-by-Step Resolution: Surgical Interactive Rebase with Verifications

1. Backup Your Repo (Safety First)

Before any history rewrite, create a full backup:

```
text Copy
git bundle create ../repo-backup.bundle --all
```

This bundles everything (commits, branches) into a single file you can restore from if needed (e.g., `git clone ../repo-backup.bundle restored-repo`).

2. Verify Current State

Ensure you're on the main branch and check the history:

```
text Copy
git checkout main
git status # Should show "ahead of origin/main" with no unstaged changes
git log --oneline --graph --decorate -n 5 # Note the hashes; confirm e6b58a19... is in t
```

Also, search for the secret locally to confirm it's still there:

```
git grep -i "sk-ant-api" # Replace with your exact secret prefix; this should hit the fix
```

If nothing shows, the issue might be elsewhere—stop and provide the output.

3. Start Interactive Rebase Targeting the Parent Commit

Rebase interactively from the parent of the offending commit. From the document, f0ab488 seems to be a safe base (pre-offending changes). Adjust if your `git log` shows differently.

```
text ⌂ Copy  
git rebase -i f0ab488~1 # The ~1 targets the parent of f0ab488, ensuring we can edit files
```

- This opens your editor (e.g., vim/nano) with a list of commits.
- Find the line for the offending commit (likely the first one: "pick e6b58a19...").
- Change "pick" to "edit" (or "e"). Save and exit.
- Git will pause at that commit, checking out its state for editing.

4. Manually Edit the Files (The Critical Fix Step)

Now you're at the historical point where the secret was introduced. The files are in their "dirty" state.

- Open each of the five files in your editor (e.g., VS Code, nano, or via command):

```
text ⌂ Copy  
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy-info.json  
# Repeat for the other four: settings-cc-proxy.json, settings-claude-long.json, settings-claude-short.json
```

- In each file, locate the secret on the specified lines (3, 4, or 10 per the error). Replace it entirely with a placeholder like "YOUR_ANTHROPIC_API_KEY_HERE" or "REDACTED_PLACEHOLDER". Ensure it's not detectable as a real key (no "sk-ant" prefix). Save all files.

- Verify the changes:

```
text ⌂ Copy  
git diff # Should show only the replacements in those five files  
git grep -i "sk-ant-api" # Should return NOTHING now
```

- If `git diff` shows unexpected changes or the grep still hits, double-check your edits and file paths.

5. Stage, Amend, and Continue Rebase

Commit the changes without altering the message:

```
text ⌂ Copy  
git add . # Stages all edited files  
git commit --amend --no-edit # Amends the current commit with the fixes  
git rebase --continue # Proceeds with the rebase; repeat edits if it stops on other commits
```

- If conflicts arise during `--continue`, resolve them (e.g., via `git mergetool` or manual edits), then `git add` and `--continue`.
- After completion:

```
text ⌂ Copy  
git log --oneline --graph -n 5 # Note the new hash for the amended commit (no longer f0ab488)  
git grep -i "sk-ant-api" -- --all # Double-check across history; should be clean
```

6. Force Push with Safety

Push the rewritten history:

```
text ⌂ Copy  
git push origin main --force-with-lease
```

- `--force-with-lease` is safer than plain `--force`; it aborts if remote has new commits.
- If it fails with the same secret error, inspect the GitHub rejection message for the new offending hash/files, and repeat from Step 3 targeting that.
- Success? Verify on GitHub: Check the repo history and secret scanning dashboard (Settings > Security > Secret scanning) to confirm no alerts.

If This Fails: Fallback to Strategy 2 (BFG)

If the rebase still doesn't clear it (e.g., secret in unexpected places), switch to BFG for a global sweep:

1. Download BFG: `brew install bfg` (macOS) or get the JAR from <https://rtyley.github.io/bfg-repo-cleaner/>.
2. Create a text file `secrets.txt` with the exact secret on one line (e.g., "sk-ant-api-01-abc123").
3. Mirror clone: `git clone --mirror https://github.com/jon-chun/clause-code-configs.git`.
4. Run BFG: `bfg --replace-text secrets.txt clause-code-configs.git`.
5. Clean: `cd clause-code-configs.git && git reflog expire --expire=now --all && git gc --prune=now -aggressive`.
6. Push: `git push`.
Then pull the cleaned mirror back to your local repo.

Post-Fix Best Practices

- Revoke the leaked key immediately via Anthropic's dashboard to minimize risk.
- Add `.gitignore` rules for config files with secrets (e.g., `*.json` under that dir).
- Enable GitHub's push protection permanently, but use env vars/secrets managers (e.g., GitHub Secrets) for API keys in the future.
- Test pushes on a new branch first.

This should finally unblock you—let me know the results or if you hit snags!

###CLARIFICATIONS:

```

Q1:
git log --oneline --graph -n 10
• 0b7ff3c (HEAD -> main) pass sec rule
• 7bd9355 chore(configuration): update settings and add new files
• e6b58a1 many initial folders
• f0ab488 (origin/main, origin/HEAD) Initial commit
Q2:
Yes, but I've manually changed:
a. all *.json files to start:
{
  "env": {
    "ANTHROPIC_AUTH_TOKEN": "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER",
    "ANTHROPIC_BASE_URL": "https://api.anthropic.com",
    "API_TIMEOUT_MS": "3000000",
    "ANTHROPIC_MODEL": "claude-sonnet-4-5-20250929"...
  }
b. added the suffix to *.json -> *.json.example
Q3:
I don't think I've tried bypass URL

```

⌚ Thought for 13s

Thanks for the clarifications—this confirms the setup. The [git log](#) shows a clean linear history with the offending commit as e6b58a1 (the full hash is likely e6b58a1962c22656566f9227eb8df9f450fcf91 from the original doc). Your manual changes to the placeholders and .json.example renames are great for the current state, but they only affect later commits (like 7bd9355 or 0b7ff3c)—the secret persists in the history at e6b58a1, which is why GitHub blocks the push. We'll surgically edit exactly there via rebase.

Since you haven't used the bypass, we'll aim for a true clean (history rewrite). Proceed on a backup repo copy to be safe. If the files at e6b58a1 don't match your current renamed ones, that's fine—we're editing the historical state temporarily.

Updated Step-by-Step Resolution: Targeted Interactive Rebase

1. Backup Your Repo

Create a bundle:

```
text ⚡ Copy
git bundle create ../repo-backup.bundle --all
```

2. Verify Current State

```
text ⚡ Copy
git checkout main
git status
git log --oneline --graph --decorate -n 10 # Should match what you shared
git grep -i "sk-ant-api" e6b58a1 # Confirms the secret is in e6b58a1's files
```

The grep should hit the five files in configuration/dot-claude_snapshot/macos_20251112-0912est/.

3. Start Interactive Rebase

Rebase from the parent of e6b58a1, which is f0ab488:

```
text ⚡ Copy
git rebase -i f0ab488
```

- Editor opens with:


```
pick e6b58a1 many initial folders
pick 7bd9355 chore(configuration): update settings and add new files
pick 0b7ff3c pass sec rule
```
- Change the first "pick" to "edit" (for e6b58a1). Save/exit.
- Git checks out e6b58a1's state.

4. Edit the Files at the Historical Point

The repo is now in e6b58a1's state—the files have the original secret.

- Open and edit each:

```
text ⚡ Copy
# Use your editor, e.g., code or nano
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy.info
```

Repeat for: settings-cc-proxy.json, settings-claude-long.json, settings.anthropic.json, settings.json.

- In each, find "sk-ant-api-01..." (lines 3, 4, or 10) and replace with "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER". Save.
- If the files here aren't renamed to .example yet (likely not, if renaming happened later), that's okay—focus on token replacement.
- Verify:

```
text ⚡ Copy
git diff # Shows only token changes
git grep -i "sk-ant-api" # Should return nothing
```

5. Amend and Continue

```
text ⚡ Copy
git add .
git commit --amend --no-edit
git rebase --continue
```

- This rewrites e6b58a1 with the clean files, then reapplies 7bd9355 and 0b7ff3c.

2. Revised Debug/Fix Step-by-Step

The terminal output aligns perfectly with expectations: The backup succeeded, the branch is ahead by 3 commits (e6b58a1, 7bd9355, 0b7ff3c), the log confirms the structure, and the `git grep` verifies the secret ("sk-ant-api03-..."). The extra file "settings-claude-default.json" is embedded in e6b58a1 across six files (not five as originally documented—adding "settings-claude-default.json"). This extra file explains potential persistence if overlooked. No other issues (e.g., conflicts or dirty working tree). The rebase plan remains solid, but I've revised Step 4 to explicitly include all six files and added a precise search/replace command for efficiency. Proceed in your current repo since the state is clean.

1. (No Change) Backup Your Repo

Already done successfully via the bundle—good job. If needed, you can restore later with `git clone ./repo-backup.bundle restored-repo`.

2. (No Change) Verify Current State

Already run; output confirms the secret is isolated to e6b58a1. For completeness, optionally re-run `git grep -i "sk-ant-api" -- --all` to ensure it's not in later commits (it shouldn't be, based on your prior manual changes).

3. (No Change) Start Interactive Rebase

```
text ⌂ Copy  
git rebase -i f0ab488
```

- In the editor, change "pick e6b58a1 many initial folders" to "edit e6b58a1 many initial folders". Leave the others as "pick". Save and exit.
- Git will pause at e6b58a1's state.

4. Edit the Files at the Historical Point (Revised for All Six Files)

Now at e6b58a1, the files contain the original secret. For precision, use a scriptable search/replace across all affected files, then verify manually if needed.

- Run this command to replace the secret globally in the directory (it targets only JSON files with the exact string):

```
text ⌂ Copy  
find configuration/dot-claude_snapshot/macos_20251112-0912est/ -type f -name "*.json"
```

This uses `sed` (macOS-compatible with `-i ''`) to replace the full secret string in all matching files.

- If you prefer manual edits, open each of the six:

```
text ⌂ Copy  
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy-info.  
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy.json  
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-default  
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-long.json  
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json  
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings.json
```

In each, locate the "ANTHROPIC_AUTH_TOKEN" key (likely on lines 3, 4, or 10) and replace the value with "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER". Save.

- Verify changes:

```
text ⌂ Copy  
git diff # Should show only the token replacements in those six files  
git grep -i "sk-ant-api" # Must return nothing
```

- If `git diff` shows extra changes or grep still hits, undo with `git checkout -- .` and retry the edits.

5. (No Change) Amend and Continue

```
text ⌂ Copy  
git add .  
git commit --amend --no-edit  
git rebase --continue
```

- Resolve any conflicts during `--continue` (e.g., if later commits like 7bd9355 renamed files, accept the placeholder versions).

- After rebase:

```
text ⌂ Copy  
git log --oneline --graph -n 10 # New hashes post-rewrite  
git grep -i "sk-ant-api" -- --all # Entire history should be clean
```

6. (No Change) Force Push

```
text ⌂ Copy  
git push origin main --force-with-lease
```

- If still blocked, capture the full error (new hash/files?) and share for further revision. On success, confirm on GitHub's secret scanning page.

If conflicts persist or the `sed` doesn't catch everything, fallback to BFG: Create a `secrets.txt` with the exact secret string on one line, then follow the mirror clone and BFG steps from before.

Finally, revoke that specific key on Anthropic's dashboard (it's compromised). Add `.gitignore` for sensitive JSON files moving forward. Let me know the next terminal outputs!

🔗 5 web pages

###TERMINAL:

```
> git rebase -i f0ab488
```

```

warning: unable to rmdir 'configuration/dot-claude_snapshot/macos_20251112-0912est/plugins/marketplaces/claude-code-workflows': Directory not empty
Stopped at e6b58a1... many initial folders
You can amend the commit now, with
git commit --amend
Once you are satisfied with your changes, run
git rebase --continue
> ls
configuration debugging LICENSE observability README.md tech-specs tutorials
> find configuration/dot-claude_snapshot/macos_20251112-0912est/ -type f -name
"*.json" -exec sed -i "" s/sk-ant-api03-
UJIG6TA9g5mSlOp46lSRonwsky3NGlei0U6Cs113aJ5j3nXKoAG5fhb5n4ixRrcDg-
xYKXhQDQudTJoVMzVw-Oti8OgAA/YOUR_ANTHROPIC_TOKEN_PLACEHOLDER/g'
{} +
> git diff # Should show only the token replacements in those six files
git grep -i "sk-ant-api" # Must return nothing
>
> git add .
git commit --amend --no-edit
git rebase --continue
[detached HEAD 7bb995a] many initial folders
Date: Thu Nov 13 00:30:56 2025 -0500
788 files changed, 241947 insertions(+)
create mode 100644 configuration/CLAUDE-github-workflow.md
create mode 100644 configuration/dot-claude_snapshot/macos_20251112-0912est/commands/issue.md
create mode 100644 configuration/dot-claude_snapshot/macos_20251112-0912est/config/notification_states.json
create mode 100644 configuration/dot-claude_snapshot/macos_20251112-0912est/debug/096e5236-2d75-4746-b723-619bb818b216.txt
create mode 100644 configuration/dot-claude_snapshot/macos_20251112-0912est/debug/11dc5cda-505f-43c1-9b89-05e871aa2cde.txt
...
create mode 100644 debugging/login-auth/login-auth_grok4_20251112.md
create mode 100644 debugging/setup-reinstall/reinstall.md
create mode 100644 debugging/shell-terminals/fonts_grok4_20251112.md
create mode 100644 observability/cc_costs.md
create mode 100644 tech-specs/cc-cli_vs_cc-vscode-ext.md
create mode 100644 tutorials/git-flow-labeler/git-flow-labler.md
create mode 100644 tutorials/git-fork-clone-subtree.md
Auto-merging configuration/dot-claude_snapshot/macos_20251112-0912est/setting.json
CONFLICT (content): Merge conflict in configuration/dot-claude_snapshot/macos_20251112-0912est/setting.json
Auto-merging configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy-info.json
CONFLICT (content): Merge conflict in configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy-info.json
Auto-merging configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy.json
CONFLICT (content): Merge conflict in configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy.json
Auto-merging configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-default.json
CONFLICT (content): Merge conflict in configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-default.json
Auto-merging configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-long.json
CONFLICT (content): Merge conflict in configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-long.json
Auto-merging configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json
CONFLICT (content): Merge conflict in configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json
error: could not apply 7bd9355... chore(configuration): update settings and add new files
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 7bd9355... chore(configuration): update settings and add new files
❶ ❷ ❸ ❹ -/code/claude-code-configs ❺ on ❻ @7bb995a8 *1 rebase-I 2/3 ~6 +11 ❻

```

⌚ Thought for 30s

The terminal shows the rebase has paused due to merge conflicts in the six JSON files when applying commit 7bd9355 ("chore(configuration): update settings and add new files") on top of your amended commit 7bb995a (the cleaned version of e6b58a1). This is expected: The amended commit replaced the secret with a placeholder in those files, but 7bd9355 likely made overlapping changes (e.g., your manual placeholder updates and/or renames to json.example), causing Git to flag content conflicts on the "ANTHROPIC_AUTH_TOKEN" lines. The "unable to rmdir" warning is likely a leftover from prior directory cleanup (e.g., nested .git issues) and can be ignored for now, as it doesn't block the rebase. Your sed replacement worked (assuming git grep returned nothing), and the amend succeeded—now we just need to resolve these conflicts manually to continue. [docs.github.com](#) +2 more

Revised Debug/Fix Step-by-Step (Continuing from Conflict State)

You're already in the rebase process (at step 2/3, on detached HEAD 7bb995a). Do NOT run `git rebase -i` again—focus on resolution. Proceed carefully to avoid losing changes.

1. Identify and Inspect Conflicts

List the conflicting files:

text	Copy
git status	

This should show "Unmerged paths" with the six files:

- configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy-info.json
- configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy.json
- configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-default.json
- configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-long.json
- configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json
- configuration/dot-claude_snapshot/macos_20251112-0912est/settings.json (note: output had a typo as "setting.json")

If other files show up, note them too.

2. Resolve Conflicts in Each File

Open each conflicting file in your editor (e.g., VS Code via `code`, or nano/vim). Look for Git's conflict markers:

- `<<<<< HEAD`: Content from the current base (your amended 7bb995a, with the new placeholder).
- `=====`: Separator.
- `>>>>> 7bd9355...`: Content from the incoming commit (7bd9355, likely with your manual changes/renames).

For each file:

- Edit to remove the markers and keep the desired final version. Since the goal is a clean history:
 - Retain the "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER" (from either side—both should have it now).
 - If 7bd9355 introduced other updates (e.g., model name changes like "claude-sonnet-4-5-20250929" or renames to `.json.example`), prefer those to preserve your later work.
 - If the file was renamed in 7bd9355 (e.g., to `.json.example`), resolve the content here—the rename will apply after resolution if Git detects it.
- Example resolution for a file like `settings.json` (simplified):

Original conflict:

```
text x ⌂ Copy
{
  "env": {
    <<<<< HEAD
    "ANTHROPIC_AUTH_TOKEN": "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER",
    =====
    "ANTHROPIC_AUTH_TOKEN": "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER", // Maybe with other
    "ANTHROPIC_MODEL": "claude-sonnet-4-5-20250929",
    >>>>> 7bd9355...
    "ANTHROPIC_BASE_URL": "https://api.anthropic.com",
```

Resolved version (keep incoming changes if better):

```
text x ⌂ Copy
{
  "env": {
    "ANTHROPIC_AUTH_TOKEN": "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER",
    "ANTHROPIC_MODEL": "claude-sonnet-4-5-20250929",
    "ANTHROPIC_BASE_URL": "https://api.anthropic.com",
```

- Save each file after editing.
- Command to open all at once (if using code):

```
text x ⌂ Copy
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings*.json
```

- After edits, verify no secrets remain:

```
text x ⌂ Copy
git grep -i "sk-ant-api"
```

Should return nothing.

3. Stage Resolved Files

Add each resolved file (do NOT commit yet—rebase handles that):

```
text x ⌂ Copy
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy-info.json
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings-cc-proxy.json
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-default.json
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings-claude-long.json
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings.json
```

Or, if only these are conflicting:

```
text x ⌂ Copy
git add .
```

Then check:

```
text x ⌂ Copy
git status # Should show no unmerged paths, and staged changes
```

4. Continue Rebase

```
text x ⌂ Copy
git rebase --continue
```

- This applies the resolved 7bd9355, then auto-applies the final commit 0b7ff3c ("pass sec rule")—it should go cleanly unless more conflicts.

- If more conflicts arise (unlikely), repeat steps 2-3 for them.
 - On success, you'll see output like "Successfully rebased and updated refs/heads/main."
 - Verify post-rebase:

```
git log --oneline --graph -n 10 # New hashes for all three commits  
git grep -i "sk-ant-api" -- --all # Entire history clean
```

5 Force Push

```
text  
git push origin main --force-with-lease
```

- If blocked again, share the full error. On success, check GitHub's secret scanning (Settings > Code security and analysis > Secret scanning) for no alerts.

If resolution gets messy (e.g., can't decide which changes to keep), abort and retry:

Then restart from the full rebase instructions, or switch to BFG for automated cleanup (as previously detailed).

Clean up the lingering directory if the `rmdir` warning bothers you post-rebase.

```
text

git rm -rf configuration/dot-claude_snapshot/macos_20251112-0912est/plugins/marketplaces/claude-
git add -u . # If it was tracked
git commit -m "Cleanup lingering directory"
```

Revoke the key on Anthropic. Share next outputs if issues!

 5 web pages

```
> git status # Should show no unmerged paths, and staged changes
interactive rebase in progress; onto f0ab488
Last commands done (2 commands done):
    edit c6b58a1 many initial folders
    pick 7bd9355 chore(configuration): update settings and add new files
Next command to do (1 remaining command):
    pick 0b7ff3c pass sec rule
    (use "git rebase --edit-todo" to view and edit)
You are currently rebasing branch 'main' on 'f0ab488'.
    (all conflicts fixed: run "git rebase --continue")
Changes to be committed:
    (use "git restore --staged <file>..." to unstage)
        new file: configuration/dot-claude_snapshot/.DS_Store
        new file: configuration/dot-claude_snapshot/macos_20251112-0912est/.DS_Store
        renamed: configuration/dot-claude_snapshot/macos_20251112-
0912est/settings.json -> configuration/dot-claude_snapshot/macos_20251112-
0912est/setting.json
        modified: configuration/dot-claude_snapshot/macos_20251112-0912est/settings-
openai-codex.json
        modified: configuration/dot-claude_snapshot/macos_20251112-0912est/settings-
zai.json
        modified: configuration/dot-claude_snapshot/macos_20251112-0912est/settings-
zai_min.json
        modified: configuration/dot-claude_snapshot/macos_20251112-0912est/settings-
zai_max.json
        modified: configuration/dot-claude_snapshot/macos_20251112-0912est/settings-
anthropic.json
        modified: configuration/dot-claude_snapshot/macos_20251112-
0912est/settings.minimax.json
        modified: configuration/dot-claude_snapshot/macos_20251112-
0912est/settings.zai.json
        modified: debugging/login-auth/login-auth_grok4_20251112.md
        renamed: tutorials/git-flow-labeler/git-flow-lablر.md -> tutorials/git-flow-
labeler/git-flow-lablر
    > git rebase --continue
[detached HEAD 29d228b] chore(configuration): update settings and add new files
11 files changed, 35 insertions(+), 8 deletions(-)
create mode 100644 configuration/dot-claude_snapshot/.DS_Store
create mode 100644 configuration/dot-claude_snapshot/macos_20251112-
0912est/.DS_Store
rename configuration/dot-claude_snapshot/macos_20251112-0912est/[settings.json]=
[setting.json] (100%)
rename tutorials/git-flow-labeler/git-flow-lablر.md => git-flow-lablر (100%)
Auto-merging configuration/dot-claude_snapshot/macos_20251112-
0912est/settings.anthropic.json.example
CONFLICT (content): Merge conflict in configuration/dot-
claude_snapshot/macos_20251112-0912est/settings.anthropic.json.example
error: could not apply 0b7ff3c... pass sec rule
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
```

The terminal output indicates progress: The rebase has advanced to the final commit (3/3), with the new commit message added as committed by "gg (GCC) Name" in line 22 of file 11.

`settings.anthropic.json.example` when applying `0b7ff3c`. This is typical for rebases involving file modifications across commits.

Revised Debug/Fix Step-by-Step (Final Resolution)

Continue directly in your current rebase state

1. Inspect the Conflict

Confirm the only unmerged path is configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json.example.

2. Resolve the Conflict

Open the file:

text  Copy
code configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json.ex

Remove conflict markers (<<<<<, =====, >>>>>) and merge content, keeping "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER" and any other desired changes from 0b7ff3c. Save. Verify:

```
text Copilot

git diff # Shows your resolutions
git grep -i "sk-ant-api" # Empty
```

3. Stage and Continue

text  Copy
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json.
git rebase --continue

This completes the release.

A Multicenter Study

```
text                                     Ⓛ Copy

git log --oneline --graph -n 10
git grep -i "sk-ant-api" -- --all
git push origin main --force-with-lease
```

If push blocks, share error. Otherwise, done—check GitHub secrets scanning

```
Step #2:  
diff -cc configuration/dot-claude_snapshot/macos_20251112-  
0912est/settings.anthropic.json.example  
index ef6543a;18e8d34..0000000  
--- a/configuration/dot-claude_snapshot/macos_20251112-  
0912est/settings.anthropic.json.example  
+++ b/configuration/dot-claude_snapshot/macos_20251112-  
0912est/settings.anthropic.json.example
```

8 Thought for 17s

The provided diff header confirms the conflict is in `settings.anthropic.json.example`, but it's incomplete (it doesn't show the actual conflicting content with markers like `<<<<<`, `=====`, `>>>>>`). To resolve, you'll need to edit the file directly, as Git has inserted those markers into the file itself during the conflict.

Continuing Resolution (From Your Current Rebase State)

1. (Completed) Inspect the Conflict

Your `git status` already confirmed the single conflicting file.

3. Resolve the Conflict

Open the file to see the full conflicted content.

text

code configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json.ex

(Or use `cat` or `less` in terminal to view: `cat configuration/dot-`

```
Inside, you'll see something like this (based on typical JSON conflicts; adjust to what you see):  
  
text  
x ⌂ ⌂ Copy  
  
{  
  "env": {  
    <<<<< HEAD # From the current base (29d228b, your resolved prior commit)  
    "ANTHROPIC_AUTH_TOKEN": "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER",  
    "ANTHROPIC_BASE_URL": "https://api.anthropic.com",  
    ===== # Separator  
    "ANTHROPIC_AUTH_TOKEN": "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER", # From incoming 0b7ff3c,  
    "ANTHROPIC_BASE_URL": "https://api.anthropic.com",  
    "API_TIMEOUT_MS": "3000000", # Example of possible added/changed line in 0b7ff3c  
    >>>>> 0b7ff3c... pass sec rule # From the incoming commit  
  ...  
}
```

- Edit to remove all markers (`<<<<<`, `=>>>=`, `>>>>>` lines).
 - Merge the content: Keep "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER" (it should be the same on both sides). Prefer any additional changes from the incoming side (0b7ff3c) if they exist (e.g., new fields, updated values like model names).

- Ensure the JSON remains valid (no syntax errors).

• Save the file.

• To confirm:

```
text Copy  
git diff # Should now show a clean diff without markers (your resolved changes)  
git grep -i "sk-ant-api" # Still empty
```

3. Stage and Continue

```
text Copy  
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json  
git rebase --continue
```

This should complete the rebase successfully, updating to a new HEAD on main.

4. Verify and Push

```
text Copy  
git log --oneline --graph -n 10 # Check new history (new hashes for all commits after fix)  
git grep -i "sk-ant-api" --all # Confirm no secrets in any commit  
git push origin main --force-with-lease
```

- If the push succeeds without secret scanning errors, you're done! Verify on GitHub: Go to repo > Settings > Security > Secret scanning to ensure no active alerts.

- If still blocked, run `git push` again and share the full rejection message (it'll list any remaining offending hash/files).

If the file's conflict is more complex than expected (e.g., no placeholders on one side), share the full content of the conflicted file (via `cat` output) for precise resolution advice. Remember to revoke the original key on Anthropic's site.

Post-fix: Commit any cleanup (e.g., for the rmdir warning from earlier) and add `.gitignore` entries like:

```
text Copy  
configuration/dot-claude_snapshot/macos_20251112-0912est/*.json  
.DS_Store
```

Then `git add .gitignore && git commit -m "Add .gitignore for secrets and macOS artifacts" && git push`.

```
####settings.anthropic.json.example
{
  "env": {
    "ANTHROPIC_AUTH_TOKEN": "YOUR_ANTHROPIC_TOKEN_PLACEHOLDER",
    "ANTHROPIC_BASE_URL": "https://api.anthropic.com",
    "API_TIMEOUT_MS": "3000000",
    "ANTHROPIC_MODEL": "claude-sonnet-4-5-20250929",
    "ANTHROPIC_SMALL_FAST_MODEL": "claude-haiku-4-5-20251015",
    "ANTHROPIC_DEFAULT_HAIKU_MODEL": "claude-haiku-4-5-20251015",
    "ANTHROPIC_DEFAULT SONNET_MODEL": "claude-sonnet-4-5-20250929",
    "ANTHROPIC_DEFAULT_OPUS_MODEL": "claude-sonnet-4-5-20250929"
  },
  "alwaysThinkingEnabled": true
}
```

⌚ Thought for 35s

The provided JSON for `settings.anthropic.json.example` looks correctly resolved: The auth token is a safe placeholder, there are no detectable secrets, and the structure is valid JSON with model configurations (though note that as of the current date, Anthropic's official models are typically named like `claude-3-sonnet-20240628` or `claude-3-haiku-20240307`—these appear to be custom or projected future variants, but that's fine for your config). If this is your merged version after removing conflict markers, proceed.

Continuing Resolution (Final Steps)

1. (Completed) Resolve the Conflict

You've edited the file to the clean JSON above—verified no secrets via grep.

2. Stage the Resolved File

```
text Copy  
git add configuration/dot-claude_snapshot/macos_20251112-0912est/settings.anthropic.json
```

3. Continue Rebase

```
text Copy  
git rebase --continue
```

- This applies the final commit (a new hash for 0b7ff3c) and completes the rebase back to main.

- If any unexpected errors, share the output.

4. Verify Post-Rebase State

```
text Copy  
git status # Should be clean, on main, ahead of origin  
git log --oneline --graph -n 10 # Confirm new linear history with updated hashes  
git grep -i "sk-ant-api" --all # Must be empty across all commits
```

5. Force Push to GitHub

```
text Copy
```

```
git push origin main --force-with-lease
```

- This rewrites the remote history to your cleaned local version.
- If successful (no secret scanning block), confirm on GitHub: Repo > Settings > Security > Secret scanning—no alerts for this key.
- If blocked again, share the full error message (it'll specify the hash/files).

You're at the finish line—this should finally allow the push. Post-push, revoke the original leaked key on Anthropic's dashboard if not already done, and consider committing a `.gitignore` update to exclude real `.json` files (keep only `.example` templates). Let me know the terminal output from the continue/push!

In VSCode file explorer the filename "settings.anthropic.json.example" is red with a "!" on the right side (does this indicate a git conflict even though the contents appear clean - could it mean there is some polluted git history/version)

⚠️ Message limit reached

Sign up for SuperGrok

Enjoy higher usage limits and more when you upgrade

- ⚡ Access to Grok 4 and new features
- ⌚ More queries for Grok 3
- ⓘ Increased memory context
- ⚡ Increased access to image generation and voice mode

[Get Started](#)

ⓘ How can Grok help?

⌚ Expert