

How to install git client-side hooks

Overview:

The purpose of these hooks is to deny commits to GitHub unless there is a valid, active Jira ticket within your commit message. The ticket should follow this format: XXX-###. Letters in all capital, followed by a hyphen (-) and the ticket number. (e.g. JIRA-1234 or WAT-253)

Your commit may also be denied if you are trying to alter a file you are not authorized/supposed to alter. You can find the permissions config containing each restricted file and the Github groups authorized to alter them [here](#). Please look at the first common issue if you have any problems with the file authorization.

Instructions

1. Ensure that python 3.0 or above is installed on your system, as well as pip. Please watch [this video](#) to install it, and make sure you include pip in the installation, as shown in the video. The python download link is [here](#). You can test the installation by running

```
1 python --version
2 pip --version
```

If this doesn't work, try moving the PATH environmental variable to the very top of all of your environmental variables. It wont interfere with anything else. If it still doesn't work, use Google 😊

2. Then install pre-commit onto your system using pip:

```
1 pip install pre-commit
```

3. Next, [clone](#) (dont download zip) the '[infr-git-hooks-clientside](#)' repository off of github and place it into the directory where you hold all of your projects. If you have a unique file structure (e.g. you have more than just GitHub/* directory holding all of your repos) then please see the disclaimer below.

i DISCLAIMER: If your local file structure does not simply hold all of the repositories you clone in one directory (e.g. Users/name/Documents/GitHub holds all of my cloned repos), you will need to edit the .pre-commit-config.yaml file as well as a filepath within the scripts. See the first common issue for guidance. You can do this after finishing the installation.

4. Then, open up a terminal / command prompt and enter the following command if you are on Windows: (You can make this any path you like, it will create the file for you. Homepath is easy, though.)

```
1 pre-commit init-templatedir C:/Users/yourname/.git-template -c /path/to/.pre-commit-config.yaml
```

If you are on Mac/Linux:

```
1 pre-commit init-templatedir ~/.git-template -c /path/to/infr-git-hooks-clientside/.pre-commit-config.yaml
```

You should have seen a message that looks like this confirming your installation:

```
1 pre-commit installed at (homepath)/.git-template/hooks/commit-msg
2 pre-commit installed at (homepath)/.git-template/hooks/pre-commit
```

5. Next, you need to set your git init template directory to this filepath. You can do this by running this command on Windows:

```
1 git config --global init.templateDir C:/path/to/.git-template
```

Or if you are on a UNIX system (Mac/Linux)

```
1 git config --global init.templateDir ~/.git-template
```

a. To ensure that this is working, go to an existing repository, and enter the command

```
1 git init
```


You should see a message such as:

```
1 Reinitialized existing Git repository in (.../.../repository/git)
```

If you see that message with no warning, try to commit some code and see if the hooks run. If you do see a warning like what's below, you will need to adjust the `init.templateDir` path to resolve it.

```
1 warning: templates not found in (.../.../.git-template)
```

Look at the beginning of step 5 and run the command again with an absolute path to `.git-template` to resolve the issue.

 **DISCLAIMER:** The first time you run these hooks on any repository, it will take a few minutes because pre-commit needs to install the required dependencies on a local environment.

DISCLAIMER: These hooks will ONLY install when you clone a new github repository or run 'git init' on a repository you already have. You can automatically install these hooks on all repos within a directory by following the guide below.

How to automatically install/uninstall hooks on all repos:

- To install, run the "install_on_all_repos.py" script within `infr-git-hooks-clientside`.
- The way this script works is it asks you to specify a parent directory in which the script will go through all subdirectories and run 'git init' on them, subsequently installing the hooks on each repo.
- To uninstall them, run "uninstall_on_all_repos.py" and provide a parent directory. It's the exact same script except it uninstalls rather than installs.

Uninstallation Guide:

If you ever want to uninstall hooks on a specific repository, simply `cd` to the desired repo, and run the following command:

```
1 pre-commit uninstall -c path/to/.pre-commit-config.yaml
```

To re-install them, just do:

```
1 pre-commit install -c path/to/.pre-commit-config.yaml
```

If you want to turn off the automatic installation, enter the following command:

```
1 git config --global --unset init.templateDir
```

To turn it back on, simply assign the `templateDir` again as shown in step 5. If you turn off automatic installation, you will still be able to install the hooks manually at any time by running the install command given before the 'git config' command.

Common Issues:

- If you are experiencing file-not-found errors when trying to run the hooks, you will need to change the `.pre-commit-config.yaml` file alongside a local filepath within the hooks code. To solve this, first open `.pre-commit-config.yaml`.

```

1  default_install_hook_types: ['commit-msg', 'pre-commit']
2
3  repos:
4    - repo: local
5      hooks:
6        - id: jira-commit-msg
7          name: JIRA Ticket Commit-Msg Check
8          entry: python ../infr-git-hooks-clientside/hooks/jira_ticket_check.py
9          language: python
10         additional_dependencies: ['requests', 'jira']
11         stages: [commit-msg]
12        - id: file-auth-pre-commit
13          name: File Permission Pre-Commit Check
14          entry: python ../infr-git-hooks-clientside/hooks/file_permission_check.py
15          language: python
16          additional_dependencies: ['requests']
17          stages: [pre-commit]

```

Find the lines of code within the red boxes, and change the path to be correct and lead to the 'infr-git-hooks-clientside' directory. The changes you make will likely want to be after the ../.

- Next, you will want to open the hooks python files (file_permission_check.py & jira_ticket_check.py)

```

1  import fnmatch
2  import re
3  import subprocess
4  import sys
5  import requests
6  import json
7
8  CONFIG_URL = "https://raw.githubusercontent.com/wincomplm/infr-git-hooks-config/main/permissions.json"
9  YOUR_PATH = "../infr-git-hooks-clientside/hooks/config/local_config.json"

```

At the top of each file, you will see 'YOUR_PATH=...' (shown in the red box.) You will want to make the same changes you made to the config, making the path lead to 'infr-git-hooks-clientside.'

- Ensure that your git config username is set and accurate. You can do this by opening the your settings/options within Github Desktop App, choosing 'git,' and setting your username to be your wincom github username (e.g. lbouayad-wincom) either locally or globally.
- Alternatively, you can set it via command prompt / terminal locally with:

```
1 git config --local user.name (name)
```

Or globally with:

```
1 git config --global user.name (name)
```

Be sure that this is your wincom-plm github account username (e.g. lbouayad-wincom). Otherwise you wont be seen as an authorized user even if you are a part of the authorized group.

i The way that setting your username globally works is just that for all git projects you have or will create that dont have an explicit local name set, it will default to the global one.

- Sometimes the Jira API token stops working. You will know this if you see "ERROR: Failed to get issue status for (JIRA-1234). Issue does not exist or you do not have permission to see it." for a ticket you know exists. You are able to use your own token rather than the default and input it into the config file. You can create your token [here](#). Open up the infr-git-hooks-clientside directory and put the token and your work email in the correct field within hooks/config/local_config.json.
- If you are experiencing file-not-found errors when trying to run the hooks, go into the .pre-commit-config.yaml file and update the 'entry:' field's filepath to match where you installed the hooks on your system.
- If you are experiencing file-not-found errors in the hooks and it mentions something about the

```
1 local_config.json
```

file, you need to enter the infr-git-hooks-clientside repository, and go to each script at

```
1 infr-git-hooks-clientside/hooks/(scripts)
```

This error really shouldn't happen, but in the case that it does, just find the fields at the top of both scripts where it says

```
1 YOUR_PATH = "../infr-git-hooks-clientside/hooks/config/local_config.json"
```

and change the filepath to be the absolute path to the `local_config.json`