Codebase Insights

Codebase Insights is a VSCode plugin that presents users with a high-level overview of their codebase. Its intended use cases are maintaining a system over a long period of time or for familiarizing oneself with a completely new codebase. Using this tool, users gain access to a few different views that can be used to determine things like which files are changed the most frequently, which appear most often in build failure stack traces, which files have only one author, and where each contributor currently has lines of code in production.

1. Installation

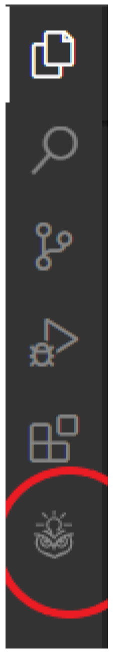
To get started, first download the latest version of the plugin and make sure that your VSCode is up to date. Then, navigate to the .vsix file from the terminal and execute the following command to install the extension.

```
code - - install-extension [path_to_extension.vsix]
```

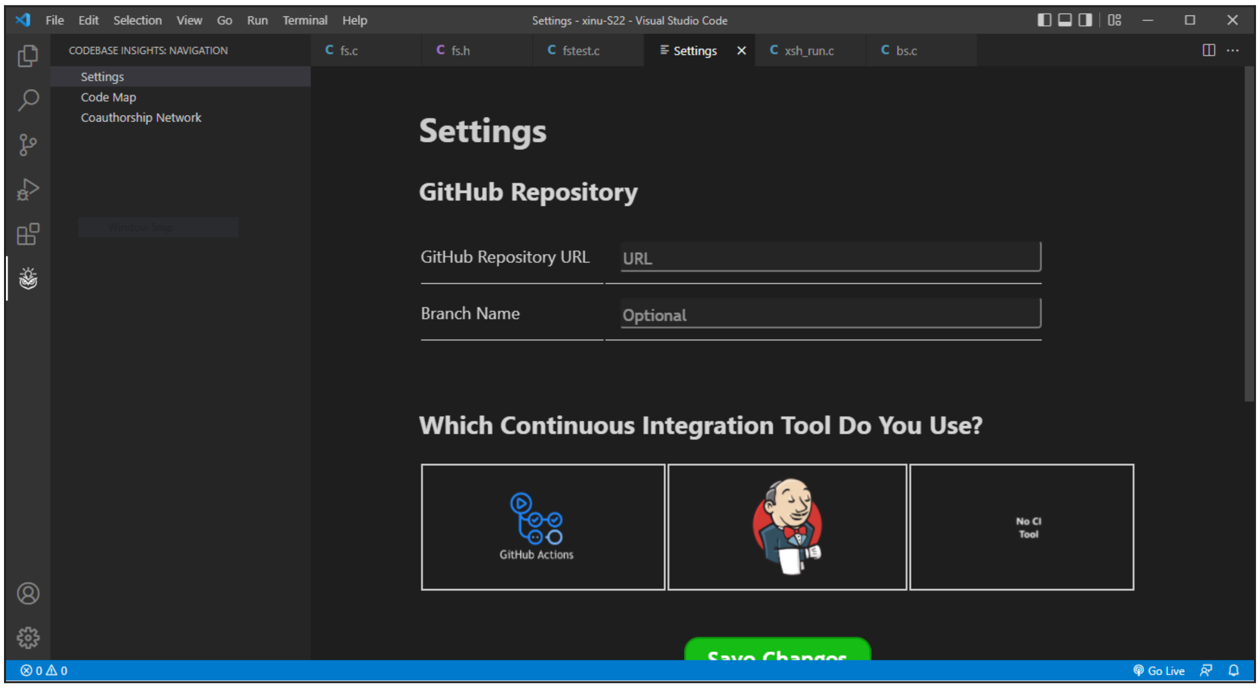
The extension should now be installed.

2. Setup

1. Launch the plugin by clicking on the Owl (located on the left side of the screen).



2. Click the Settings tab



File Edit Selection View Go Run Terminal Help
Settings - xinu-S22 - Visual Studio Code
CODEBASE INSIGHTS: NAVIGATION
Settings
Code Map
Coauthorship Network
fs.c
fs.h
fstest.c
Settings
xsh_run.c
bs.c

Settings

GitHub Repository

GitHub Repository URL: URL

Branch Name: Optional

Which Continuous Integration Tool Do You Use?

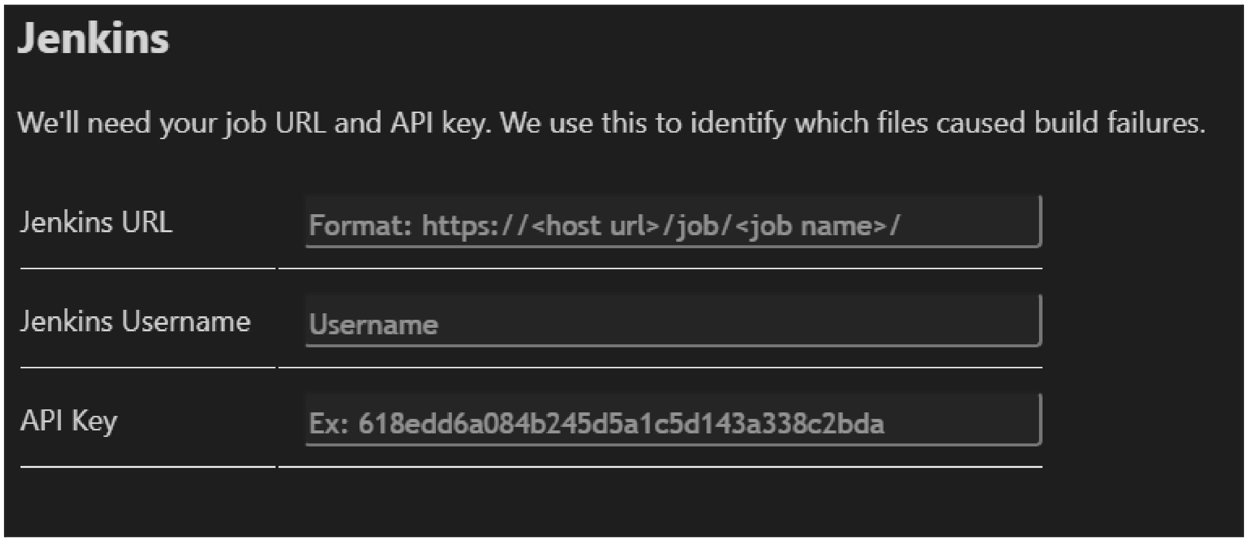
3. Enter the URL of your GitHub repository. Your URL must end with your project name.

   Example 1: https://github.com/myusername/MyProject

   Example 2: https://github.iu.edu/mygroup/MyOtherProject/

4. Click on one of the 3 images to select a CI tool. By entering information about your CI tool, you authorize Codebase Insights to analyze which build failures failed and when.

5. **If you use Jenkins**: follow the steps below. **<u>If you do not use Jenkins, skip to Step 6.</u>**
    a. You will see this menu on the Settings tab after clicking on the Jenkins logo.

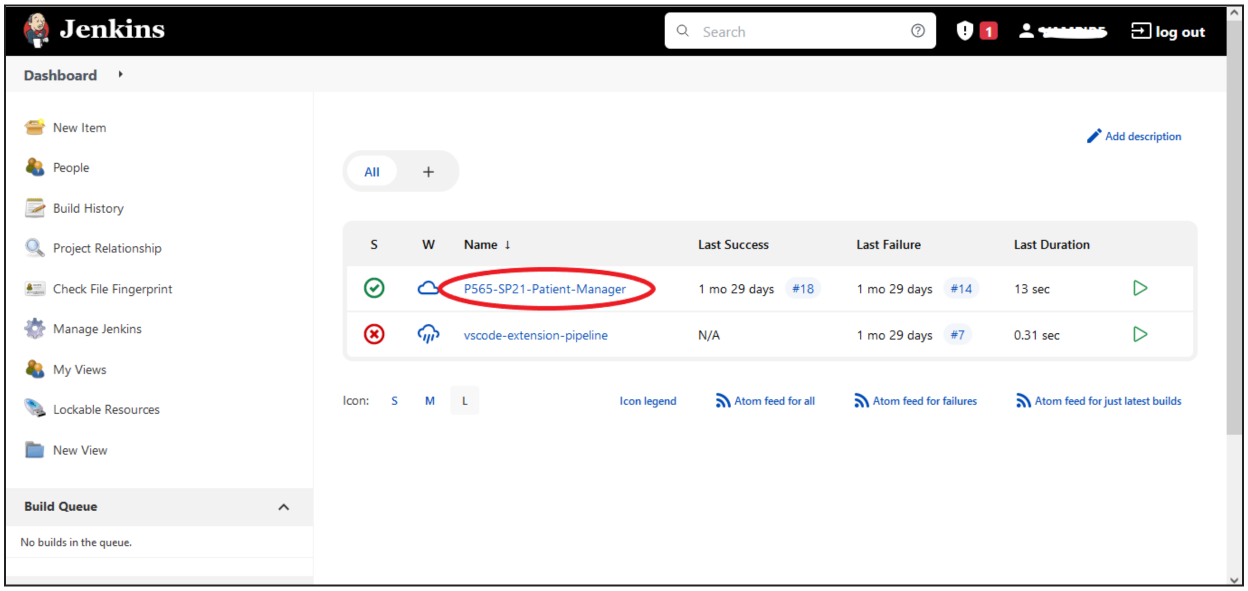


Jenkins

We'll need your job URL and API key. We use this to identify which files caused build failures.

Jenkins URL	Format: https://<host url>/job/<job name>/
Jenkins Username	Username
API Key	Ex: 618edd6a084b245d5a1c5d143a338c2bda

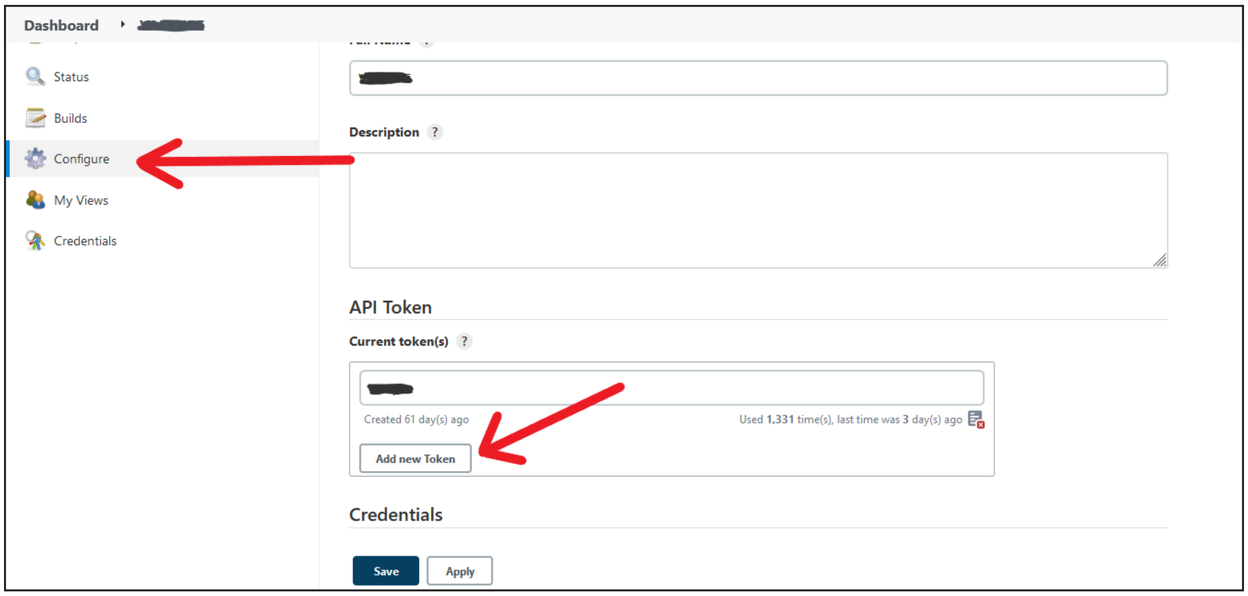
    b. **Jenkins URL**: To find this, log in to your Jenkins account and select your job. Then, copy the URL from your browser.



S	W	Name ↓	Last Success	Last Failure	Last Duration
		P565-SP21-Patient-Manager	1 mo 29 days #18	1 mo 29 days #14	13 sec
		vscode-extension-pipeline	N/A	1 mo 29 days #7	0.31 sec

    c. **Jenkins Username**: Self-explanatory

d. **Jenkins API Key**: Go to your account settings, then select Configure. Click the "Add new Token" button.



Dashboard ▸ [redacted]

- Status
- Builds
- Configure
- My Views
- Credentials

[redacted]

Description ?

API Token

Current token(s) ?

[redacted]

Created 61 day(s) ago

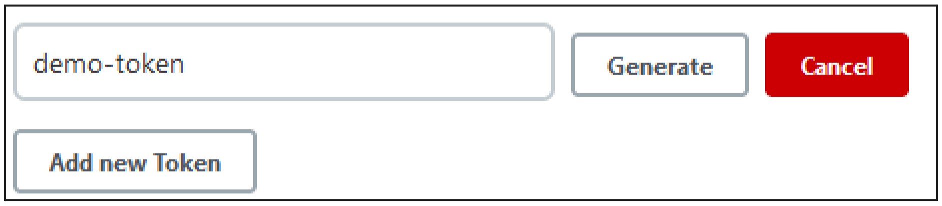
Used 1,331 time(s), last time was 3 day(s) ago

Add new Token

Credentials

Save Apply

e. Enter an identifier for your API token—it can be whatever text you choose—then click "Generate."

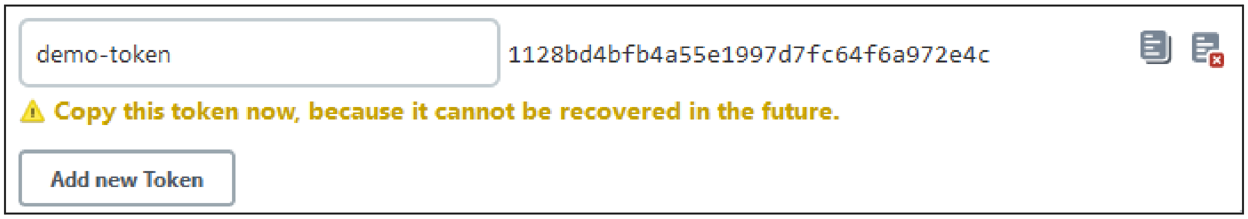


demo-token

Generate Cancel

Add new Token

f. Copy the token Jenkins generates into the API Key field into the Settings page of Codebase Insights.

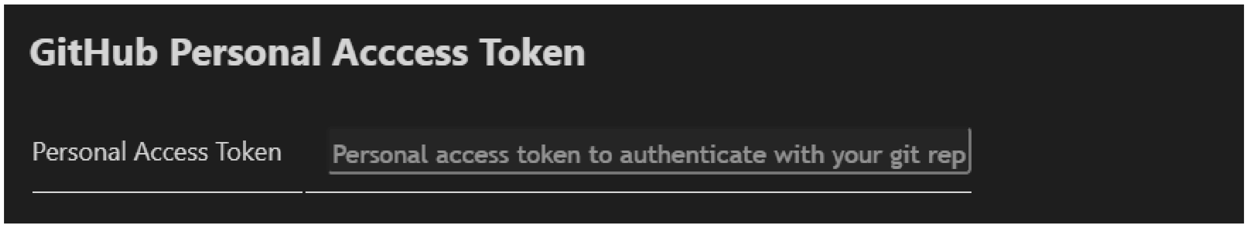


demo-token 1128bd4bfb4a55e1997d7fc64f6a972e4c

⚠ Copy this token now, because it cannot be recovered in the future.

Add new Token

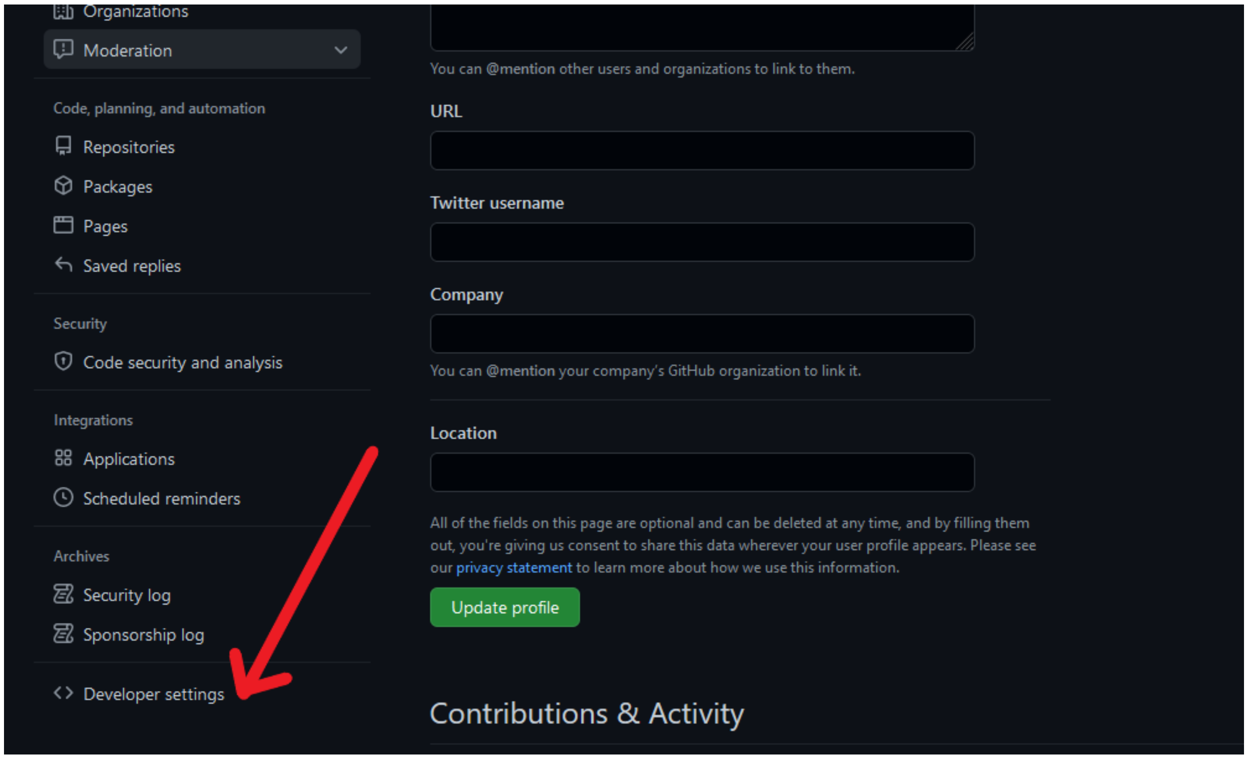
6. **GitHub Personal Access Token (PAT)**: This must be entered into the below field. To get a PAT, follow the steps below.



GitHub Personal Acccess Token

Personal Access Token | Personal access token to authenticate with your git rep

a. Open github.com or your enterprise GitHub page. Next, go your account's "Settings" page, then click on "Developer settings".

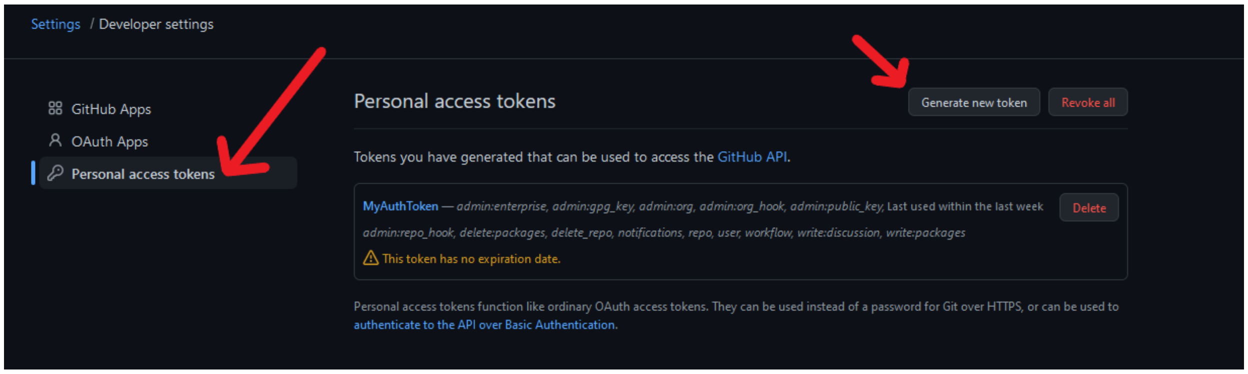


You can @mention other users and organizations to link to them.

You can @mention your company's GitHub organization to link it.

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our privacy statement to learn more about how we use this information.

b. Select "Personal access tokens" from the right, then click "Generate new token".



Tokens you have generated that can be used to access the GitHub API.

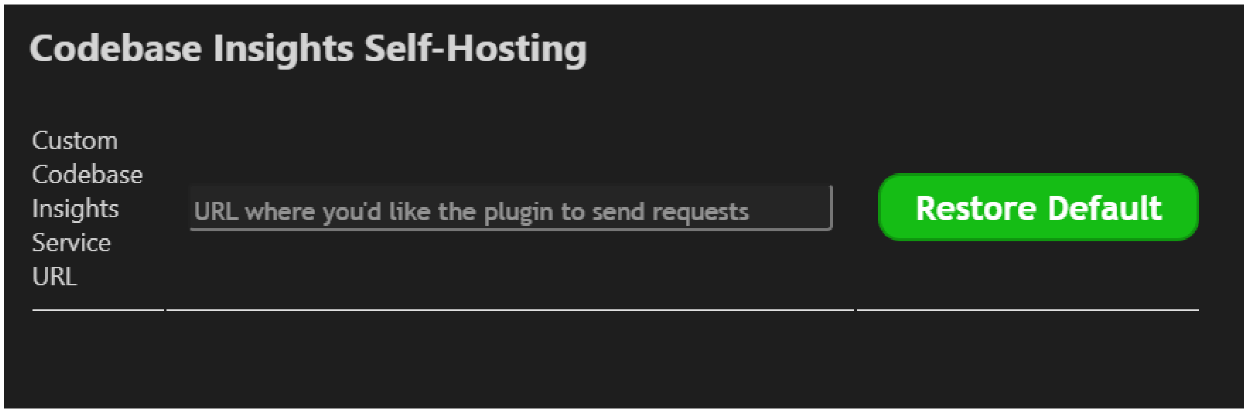
MyAuthToken — *admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key,* Last used within the last week
admin:repo_hook, delete:packages, delete_repo, notifications, repo, user, workflow, write:discussion, write:packages
This token has no expiration date.

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

c. Follow the instructions to create a token. We recommend selecting all "repo" fields and the "read:org" field for the token's Scope to grant Codebase Insights permission to clone your repository.

7. **Self-Hosting (Optional)**: We recognize that not all developers are comfortable with allowing Codebase Insights to clone their code and analyze it remotely. If you prefer to run the Codebase Insights service yourself, we provide a .jar file which launches the backend server. This server must be run each time the plugin is used with '`java -jar codebase-insights-service.jar`'. Enter the URL that can be used to access that service followed by /api.

Example 1: http://localhost:8080/api


Codebase Insights Self-Hosting
Custom Codebase Insights Service URL
URL where you'd like the plugin to send requests
Restore Default

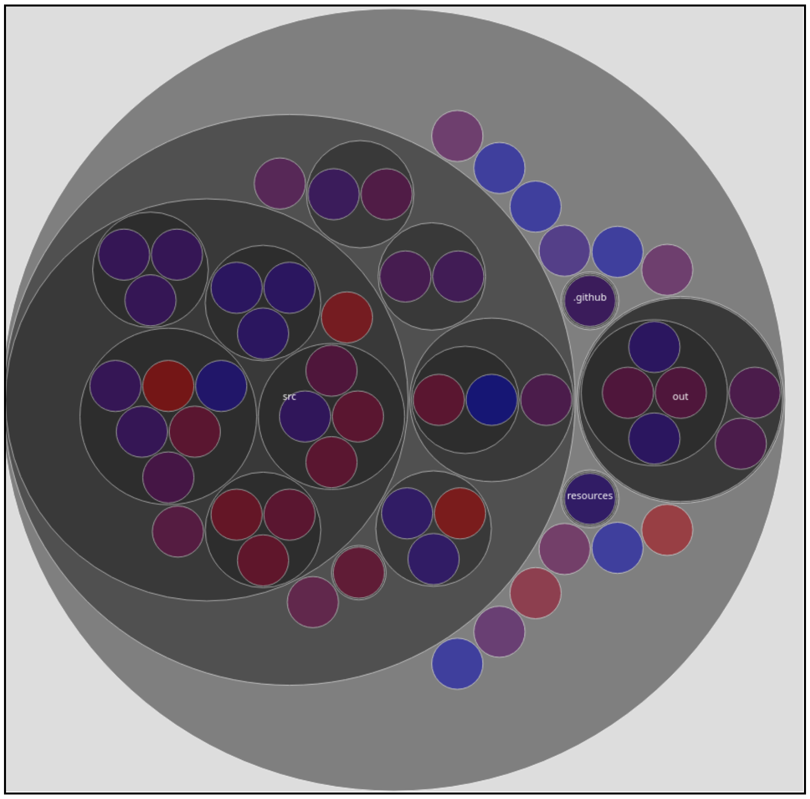
8. Finally, click the "Save Changes" button. If all goes well, you should see an "Analysis complete" message on the Settings page. The analysis may take a few minutes to complete for the first time.

3. Usage

Once you have completed the installation and setup, you can simply navigate to the code map, coauthorship network, or the commit risk assessment tabs to view the results of their analysis.

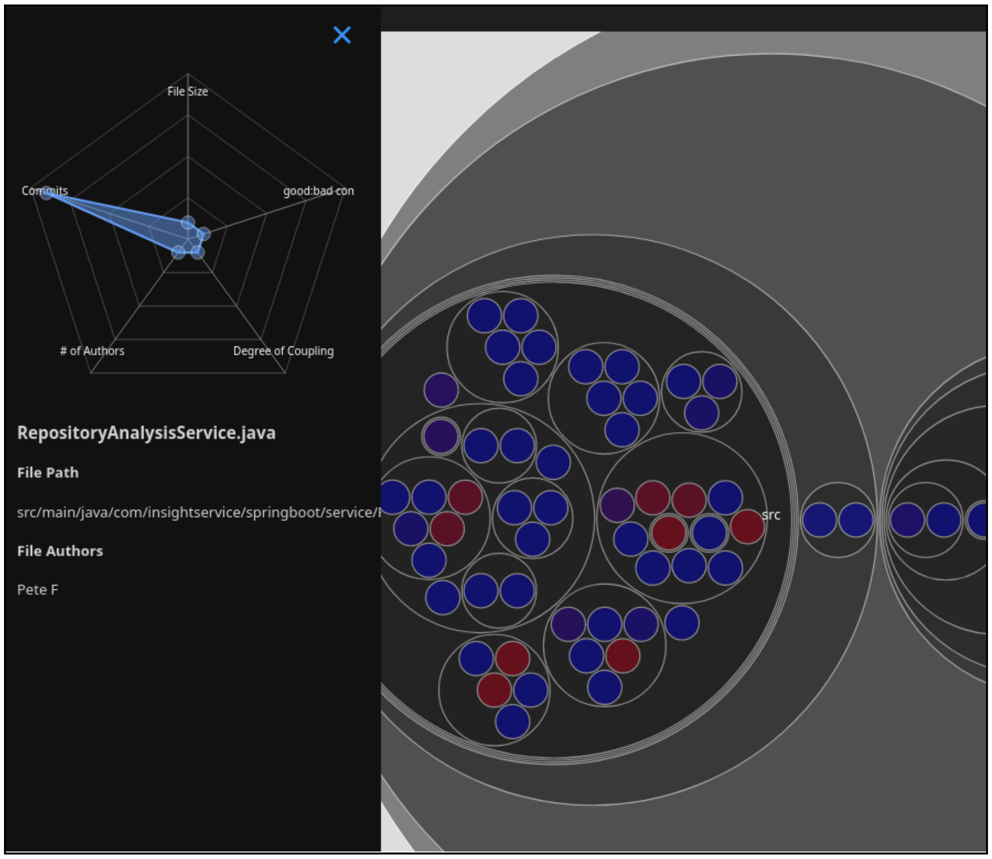
a. **Code Map**

Before proceeding, you must have followed the instructions for Setup to enter the necessary information on the Settings tab and run an analysis.
The Code Map is the flagship feature of the plugin and where you go to get a visual overview of your codebase. After clicking on the Code Map tab, you'll likely have to wait a few seconds before anything appears. If everything in the settings page was set up correctly, you'll shortly see something like this.



This map of the repository can be thought of like a file system tree. The innermost circles are individual files, and the gray circles are directories or packages. Here, each of the colored circles represents an individual file in your codebase. The ones which are more red have been marked by the analysis as having more “heat,” which is essentially a way to quantify how active a file is. Generally speaking, “hot” files are the ones which are being changed all the time by a bunch of different people, while “cold” (blue) files are the ones which have remained fairly stable for a long time.

If you click on a file, you can get a more detailed breakdown of its heat metrics in a side panel **and also see the file’s name.** This view also includes information about who are considered to be the authors of the file.



RepositoryAnalysisService.java

File Path

src/main/java/com/insightservice/springboot/service/

File Authors

Pete F

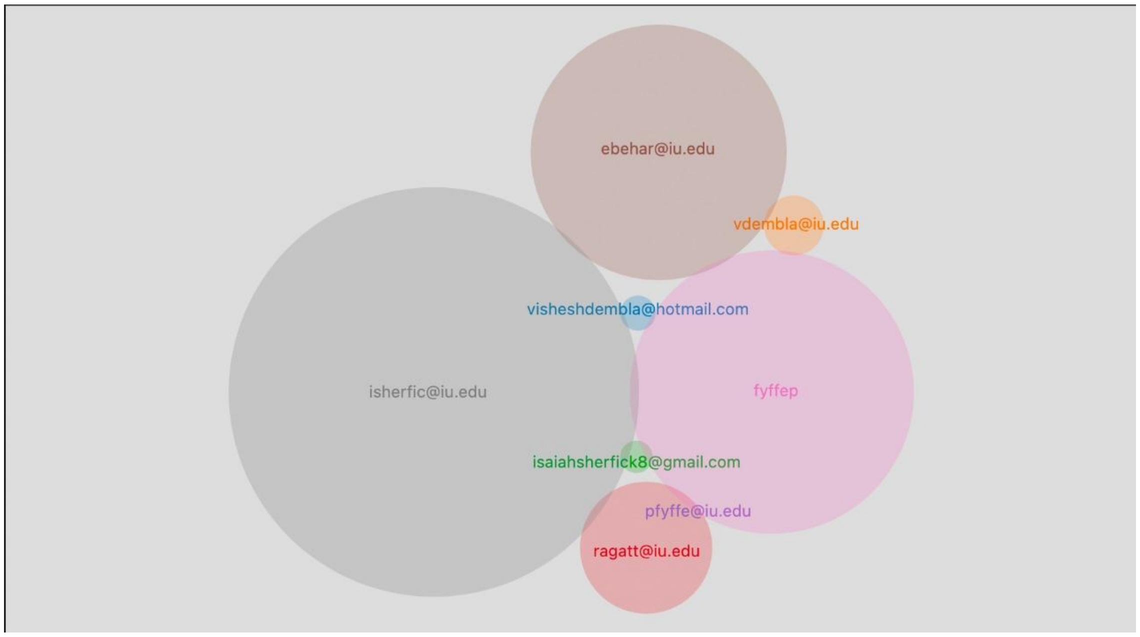
Clicking on a directory will zoom in to that directory and display the names of its child folders. Clicking on the same directory again will zoom back out to the root.

Click on the show control panel button in the top left to make the control panel appear. Here, you can isolate which heat metric you'd like to have displayed as a gradient instead of seeing the weighted sum of them all.

b. Coauthorship Network

Before proceeding, you must have configured the settings tab correctly.

In the coauthorship network, you can view the files in which each contributor has authorship on the analysis branch. Each author's circle is larger based on how many lines of code they have written.



Clicking on an author's node will show the files they have authored, sorted by **bus factor** (A file with few authors is bad (red) and a file with more authors is better (blue)). Clicking on the overlap between two authors will show the files that they share.

c. Commit Risk Assessment

To use the commit risk assessment feature, first ensure that you are running vscode *from the root of the analysis repository.* Also ensure that you are on the analysis branch. After doing so, simply stage some files for a commit with git add as you normally would, but navigate to this tab before committing. Confirm that the files you have staged are visible and click "run analysis." Wait a few seconds, and the plugin will tell you a few things about the files you've changed, such as:

- Various "heat metrics" related to each of your staged files
- A list of files highly coupled to some of your staged files which are *not* changed.

4. For Developers: Extending the Plugin

Codebase Insights is open-source and available for forking. The frontend code can be found here, and we describe how to work in that environment below. The backend code can be found here with its own section below as well. To extend this plugin to support other IDEs, it would just require a new frontend project for that given IDE's environment.

a. **<u>Frontend Development</u>**

Working with the frontend can be a little tricky because VSCode has its own custom API that you have to work around. For the most part, it's just a TypeScript project, but things get a bit tricky regarding the actual bit that makes our frontend appear: VSCode WebViews. Documentation for those, which were the main source that we read from when initially developing the plugin, can be found here. There is also a great set of articles from Microsoft on getting started with VSCode extension development here which we referenced as well.

1. **<u>WRITING TESTS</u>**
   a. All tests for the frontend are located in src/test/suite/extension.test.ts. Simply add another test method to that file and then run the test launch config to run the tests.
2. **<u>Adding a new webview</u>**
   a. *What is a webview?* Each webview represents a tab in the plugin.
   b. Add a new command for your webview to package.json alongside the other webview commands (in the "contributions" section).
   c. Add your webview to the getCommand method in treeviews/codeBaseInsightsView.ts.
   d. Add your webview to the views array in treeviews/codeBaseInsightsViewProvider.ts.
   e. Add your webview to webviews/webviewFactory.ts in the same manner as the existing ones.
   f. Ensure that the script and resource directories are added to the localResourceRoots while creating a webviewPanel.
   g. Add your webview to htmlFactory.ts in the same manner as the other ones.
   h. Create webviews/your_new_webview/your_new_webview.ts, webviews/your_new_webview/your_new_webviewScript.js, and webviews/your_new_webview/your_new_webview.css.
3. **<u>Modifying the Code Map</u>**

   a. Almost all of the code map's frontend functionality can be found in webviews/codeMap/codeMapScript.js. It was originally based on this d3.js example project and has since been heavily modified.
   b. To pass a message back to the backend, message passing must be utilized in order to invoke the methods found in src/api/api.ts. Note that this includes adding or modifying a .onDidMessageReceive() function to the code map in src/webviews/webviewFactory.ts.

4. **Modifying the Coauthorship network**
   a. Almost all of the CAN's frontend functionality is in webviews/coauthorshipNetwork/coauthorshipNetworkScript.js. It was originally based on this d3.js example project and has undergone less modification than the code map.
   b. Again, to pass a message back to the backend, message passing must be utilized to invoke the methods found in src/api/api.ts. Note this includes adding or modifying a .onDidMessageReceive() function to the CAN in src/webviews/webviewFactory.ts.
5. **Modifying the Settings page**
   a. The settings page has a lot of functionality in src/webviews/settings/settingsScript.js, but it also has a lot of functionality in src/config/config.ts. Note that adding new config fields requires also modifying src/package.json in the same manner as the current config fields have been done.
6. **Modifying the Commit Risk Assessment page**
   a. This page is handled exactly the same way as the above webviews. Its git functionality is present in src/utils/git.ts.

b. Backend Development

The backend repo is a Maven-based Spring Boot application that brings up various REST endpoints which are required for the frontend to interact with the backend. These REST based endpoints are defined in various Controller classes under the package "*com.insightservice.springboot.controller*". Based on the various webviews of the frontend, there is a corresponding controller at the backend satisfying the requirements for the same.

1. **<u>Adding a new REST endpoint:</u>**
    a. Identify the controller in which this new endpoint would go based on the top-level feature like Coauthorship Network or CodeMap.
    b. If the endpoint is part of a new feature, you can create a new controller for that in the aforementioned package.
    c. For more information about how to add a new REST endpoint in Spring you can refer to here.

2. **<u>Adding support for a new CI tool:</u>**
    a. You can add support for a new CI tool in the package "*com.insightservice.springboot.utility.ci_analyzer*".
    b. The calling hierarchy for this would be:
    RepositoryAnalysisController ⇒ RepositoryAnalysisService ⇒
NewCiService.
    c. The new class would contain the required HTTP calls to the new service based on the input parameters sent by the frontend (username, password, URL).

3. **<u>Modifying the Code Map:</u>**
    a. The controller logic for Code Map resides in the class RepositoryAnalysisController.
    b. The calling hierarchy for this would be:
    RepositoryAnalysisController ⇒ RepositoryAnalysisService

4. **<u>Modifying the Coauthorship network:</u>**
    a. The controller logic for Code Map resides in the class KnowledgeGraphController.
    b. The calling hierarchy for this would be:
    KnowledgeGraphController ⇒ RepositoryAnalysisService