

# Installation Document

## How to Integrate PostgreSQL Database to XAMPP in Windows:

We are using [XAMPP](#), so decided to just integrate PostgreSQL into it rather than compiling it with PHP from scratch.

### Installing PostgreSQL

1. Download the PostgreSQL installer from [EnterpriseDB](#).
2. Run the installer and follow the on-screen instruction.  
Note: pgAdmin also get installed alongside.
3. Assuming XAMPP is located in **C:\xampp**; using the pgSQL installer, install PostgreSQL in say **C:\xampp\pgsql\9.1** folder.
4. You will be prompted to set a password for **postgres** root user.
5. By now, pgSQL has been installed.

### Getting PostgreSQL to talk with PHP

We need to perform the following rituals to get PHP talking / communicating with pgSQL.

1. Open **php.ini** file located in **C:\xampp\php**.
2. Uncomment the following lines in *php.ini*

```
extension=php_pdo_pgsql.dll  
extension=php_pgsql.dll
```

3. Add the below code snippet to **httpd.conf**

```
Load File "C:\xampp\php\libpq.dll"
```

4. Done.

### PostgreSQL Database Administration tool

When we were installing PostgreSQL, [pgAdmin](#) – a graphical pgSQL [database administration](#) tool was installed alongside.

There is also [phpPgAdmin](#) (web base Postgres database management tool) which is to PostgreSQL what phpMyAdmin is to MySQL / MariaDB.

### pgAdmin Quick-start

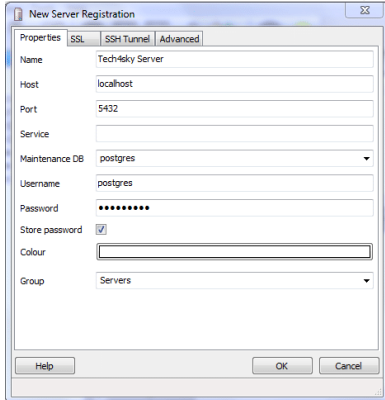
To get started with pgAdmin in creating and managing postgres database; firstly create a server, connect to it and create the database for that project of yours.

Let's start with creating the server.

1. Click on **File > Add Server...**

# Installation Document

1. Fill the form as shown in the image below.



Take note: in the username and password field, insert `postgres` as the username and in the password field, the password you entered while installing postgresSQL.

To create the database, under the server we created, right-click on the Database menu and click the **New Database...** link.

## phpPgAdmin installation

I will be integrating phpPgAdmin to XAMPP to just have a MySQL-like XAMPP experience.

1. Head over to the [Github repository](#) and clone the repo to `C:\xampp\phppgadmin`. Alternatively, download the repo. as a Zip, and extract the content to `C:\xampp\phppgadmin`.
2. In `C:\xampp\phppgadmin\conf`, rename the `config.inc.php-dist` file to `config.inc.php`
3. Edit the `config.inc.php` and replace all instances of the following with the values below.

```
4.  
5. $conf['servers'][0]['host'] = 'localhost';  
6. $conf['servers'][0]['pg_dump_path'] = 'C:\\xampp\\pgsql\\9.1\\pg_dump.exe';  
7. $conf['servers'][0]['pg_dumpall_path'] = 'C:\\xampp\\pgsql\\9.1\\pg_dumpall.exe';  
8. $conf['extra_login_security'] = false;
```

9. Edit XAMPP's `httpd-xampp.conf` and add the below code.

```
10.  
11. Alias /phppgadmin "C:/xampp/phppgadmin/"  
12. <directory "C:/xampp/phppgadmin">  
13. AllowOverride AuthConfig  
14. Require all granted  
15. </directory>
```

16. Restart Apache
17. You should now be able to use phpPgAdmin when you visit `http://localhost/phppgadmin`.

Putting pgSQL and PHP to test

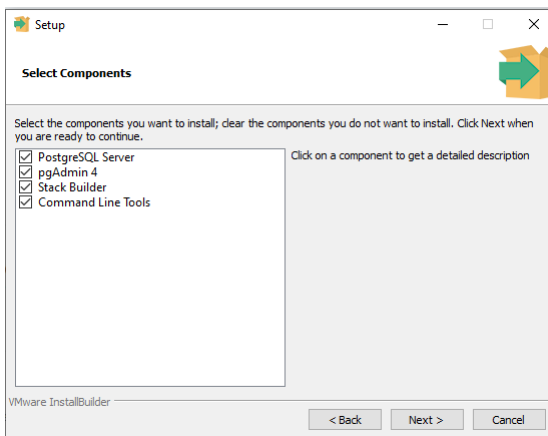
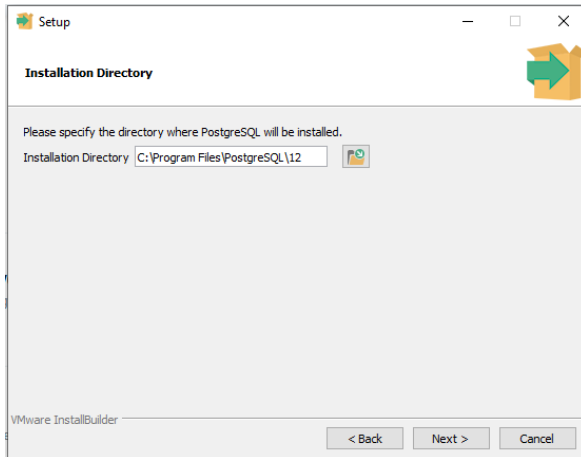
I will be creating a test that check a PHP connection to postgres using `pg_connect()`.  
If successful, PostgreSQL connection resource is returned or FALSE on failure.

```
<?php  
$link = pg_connect("host=localhost port=5432 dbname=postgres user=postgres password=mypassword");  
print_r($link);
```

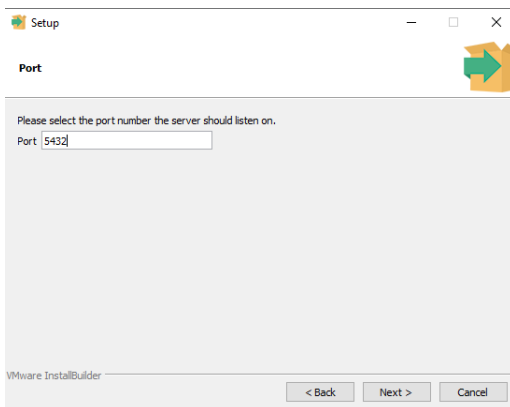
# Installation Document

## POSTGRESQL INSTALLATION:

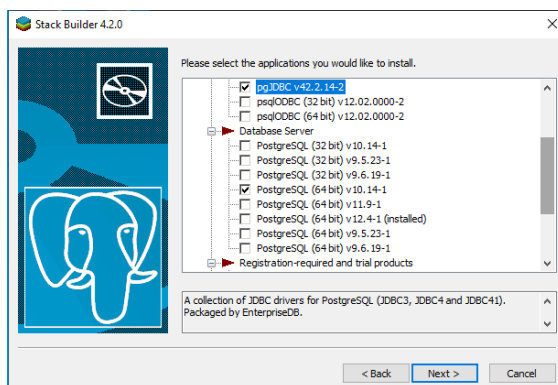
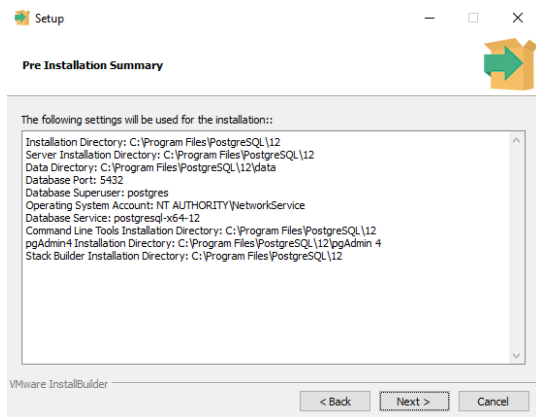
C:\Program Files\PostgreSQL\12



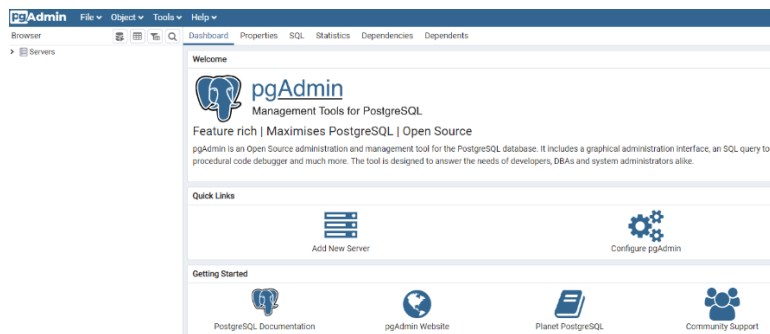
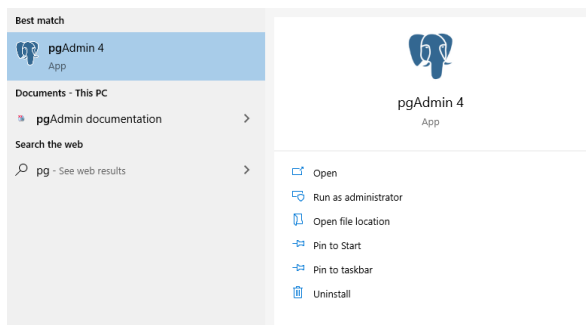
C:\Program Files\PostgreSQL\12\data



# Installation Document



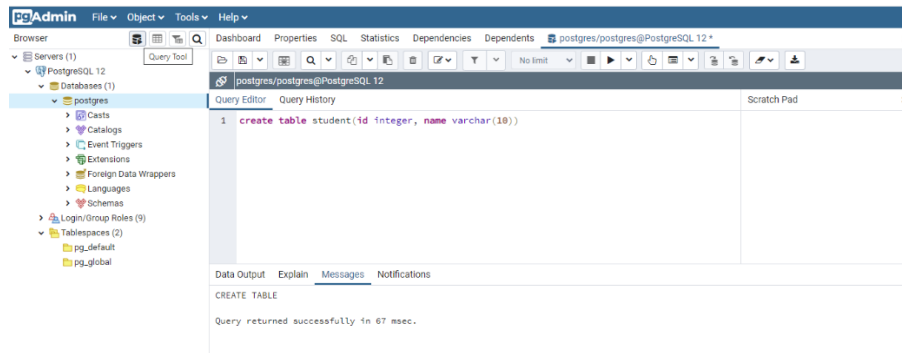
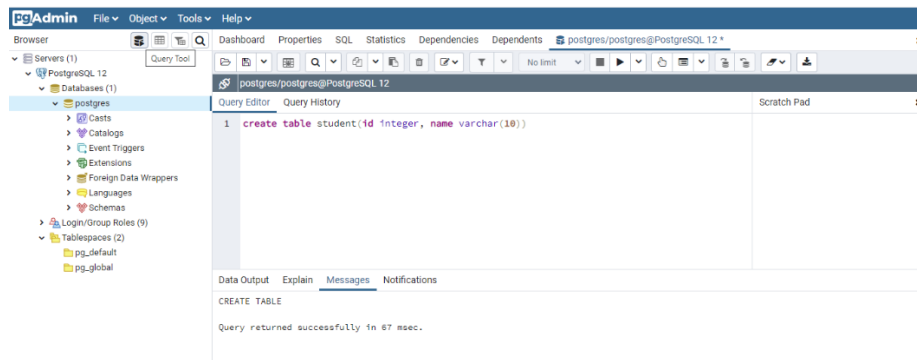
## PGADMIN:



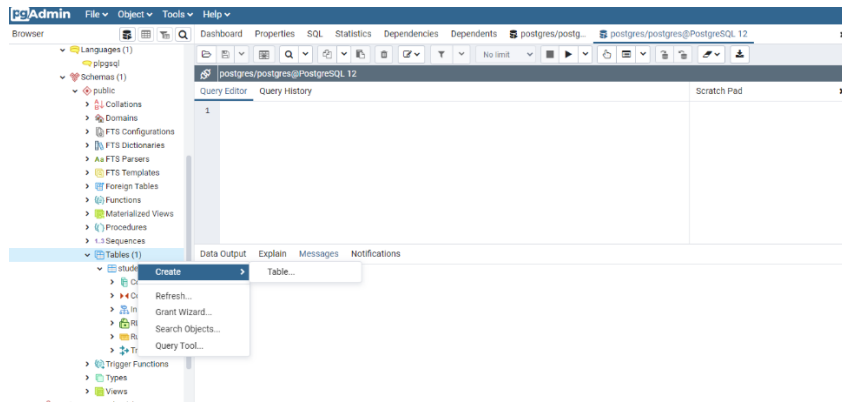
## Writing queries:

# Installation Document

You need to be on the database postgres and click on the query tool. postgres is the default database.



## Create tables with the UI:



## PSQL SHELL

Examples:

- ❓ \q: Quit/Exit
- ❓ \c \_\_database\_\_: Connect to a database
- ❓ \d \_\_table\_\_: Show table definition (columns, etc.) including triggers
- ❓ \d+ \_\_table\_\_: More detailed table definition including description and physical disk size
- ❓ \l: List databases

- \i C:/Users/Priya/Desktop/viewpostgre.sql to load a file

# Installation Document

```
SQL Shell (psql)
```

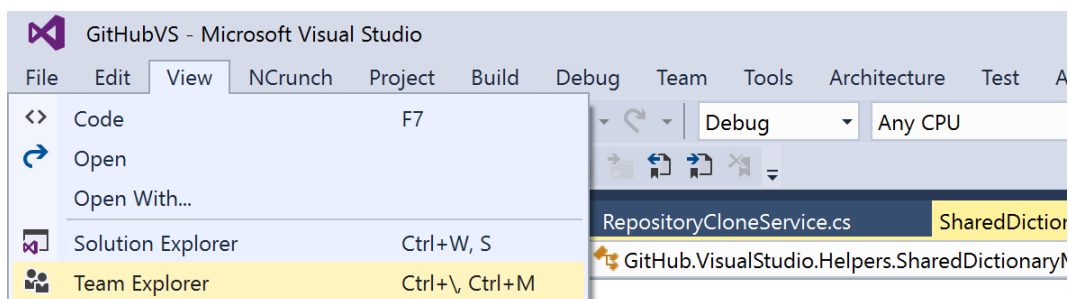
```
Server [localhost]:  
Database [postgres]:  
Port [5432]:  
Username [postgres]:  
Password for user postgres:  
psql (12.4)  
WARNING: Console code page (437) differs from Windows code page (1252)  
8-bit characters might not work correctly. See psql reference  
page "Notes for Windows users" for details.  
Type "help" for help.  
  
postgres=# \d  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
public | student | table | postgres  
(1 row)  
  
postgres=#
```

```
SQL Shell (psql)
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=# CREATE TABLE student2(id integer, lastname varchar(10));
CREATE TABLE
postgres=# SELECT * FROM "student2";
 id | lastname 
----+-----
(0 rows)
```

## Authenticating to GitHub:

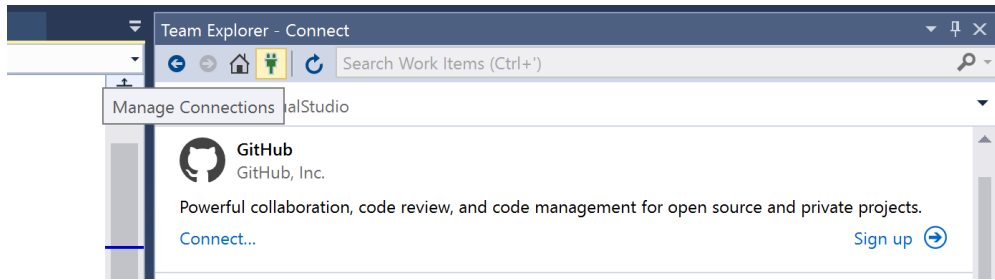
## How to login to GitHub or GitHub Enterprise

1. In Visual Studio, select **Team Explorer** from the **View** menu.

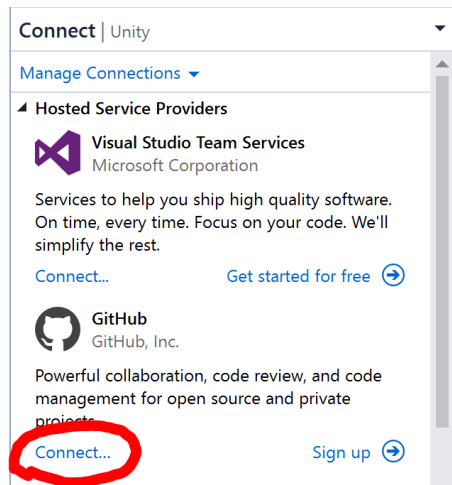


2. In the Team Explorer pane, click the **Manage Connections** toolbar icon.

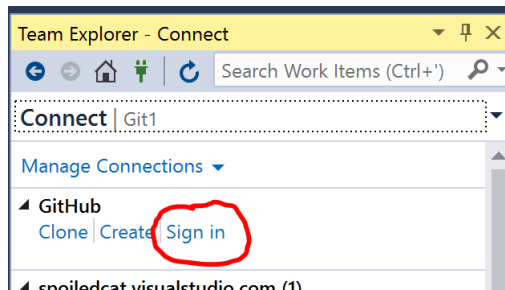
# Installation Document



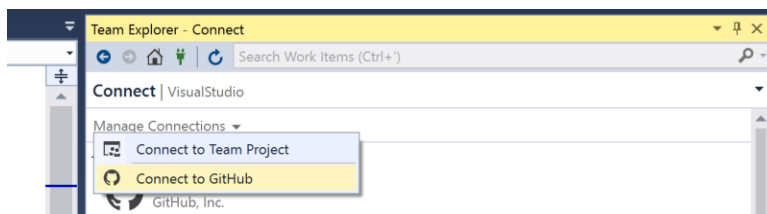
3. Click the **Connect** link in the GitHub section.



If you're connected to a TFS instance, click on the **Sign in** link instead



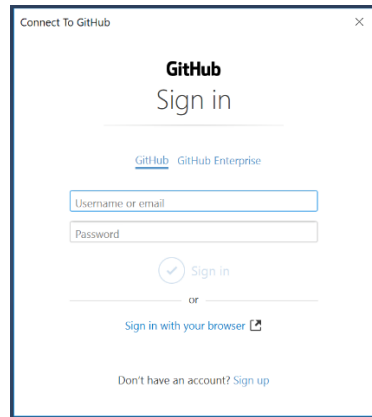
If none of these options are visible, click **Manage Connections** and then **Connect to GitHub**.



4. In the **Connect to GitHub** dialog choose **GitHub** or **GitHub Enterprise**, depending on which product you're using.

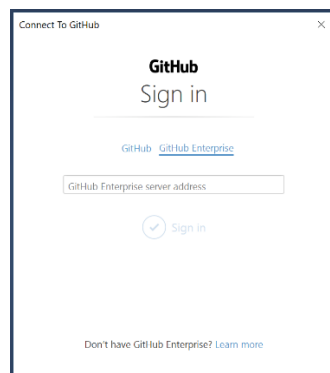
**GitHub option:**

# Installation Document

A screenshot of the GitHub 'Connect To GitHub' dialog box. It features the GitHub logo and 'Sign in' text. Below this are links for 'GitHub' and 'GitHub Enterprise'. There are two input fields: 'Username or email' and 'Password'. A 'Sign in' button with a checkmark icon is present, along with a link 'Sign in with your browser' and an external link icon. At the bottom, it says 'Don't have an account? Sign up'.

- To sign in with credentials, enter either username or email and password.
- To sign in with SSO, select Sign in with your browser.

## GitHub Enterprise option:

A screenshot of the GitHub 'Connect To GitHub' dialog box for the Enterprise option. It features the GitHub logo and 'Sign in' text. Below this are links for 'GitHub' and 'GitHub Enterprise'. There is a single input field labeled 'GitHub Enterprise server address'. A 'Sign in' button with a checkmark icon is present. At the bottom, it says 'Don't have GitHub Enterprise? Learn more'.

- To sign in with SSO, enter the GitHub Enterprise server address and select Sign in with your browser.
- To sign in with credentials, enter the GitHub Enterprise server address.
  - If a Password field appears, enter your password.
  - If a Token field appears, enter a valid token. You can create personal access tokens by [following the instructions in the section below](#).

Before you authenticate, you must already have a GitHub or GitHub Enterprise account.

- For more information on creating a GitHub account, see "[Signing up for a new GitHub account](#)".
- For a GitHub Enterprise account, contact your GitHub Enterprise site administrator.

## Personal access tokens

If all signin options above fail, you can manually create a personal access token and use it as your password.

The scopes for the personal access token are: user, repo, gist, and write:public\_key.



# Installation Document

- *user* scope: Grants access to the user profile data. We currently use this to display your avatar and check whether your plans lets you publish private repositories.
- *repo* scope: Grants read/write access to code, commit statuses, invitations, collaborators, adding team memberships, and deployment statuses for public and private repositories and organizations. This is needed for all git network operations (push, pull, fetch), and for getting information about the repository you're currently working on.
- *gist* scope: Grants write access to gists. We use this in our gist feature, so you can highlight code and create gists directly from Visual Studio
- *write:public\_key* scope: Grants access to creating, listing, and viewing details for public keys. This will allows us to add ssh support to your repositories, if you are unable to go through https (this feature is not available yet, this scope is optional)

For more information on creating personal access tokens, see "[Creating a personal access token for the command line](#)".

For more information on authenticating with SAML single sign-on, see "[About authentication with SAML single sign-on](#)".

# Installation Document

## Working with GitHub in VS Code

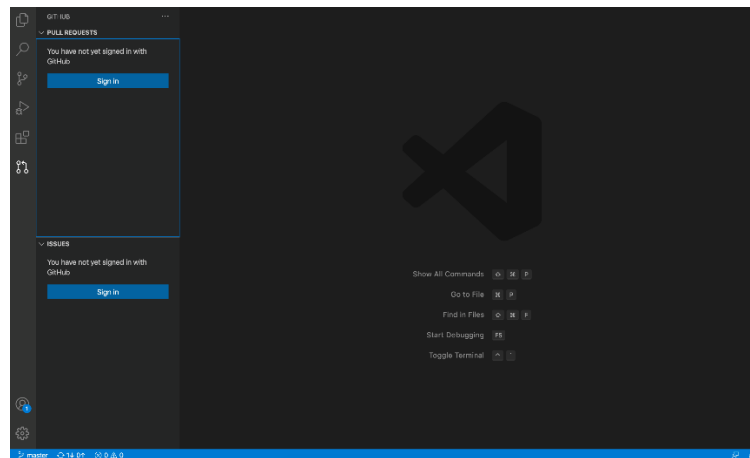
Using [GitHub](#) with Visual Studio Code lets you share your source code and collaborate with others. GitHub integration is provided through the [GitHub Pull Requests and Issues](#) extension.

To get started with the GitHub in VS Code, you'll need to [create an account](#) and install the [GitHub Pull Requests and Issues](#) extension. In this topic, we'll demonstrate how you can use some of your favorite parts of GitHub without leaving VS Code.

If you're new to source control and want to start there, you can learn about VS Code's [source control integration](#).

### Getting started with GitHub Pull Requests and Issues#

Once you've installed the [GitHub Pull Requests and Issues](#) extension, you'll need to sign in. Follow the prompts to authenticate with GitHub in the browser and return to VS Code.

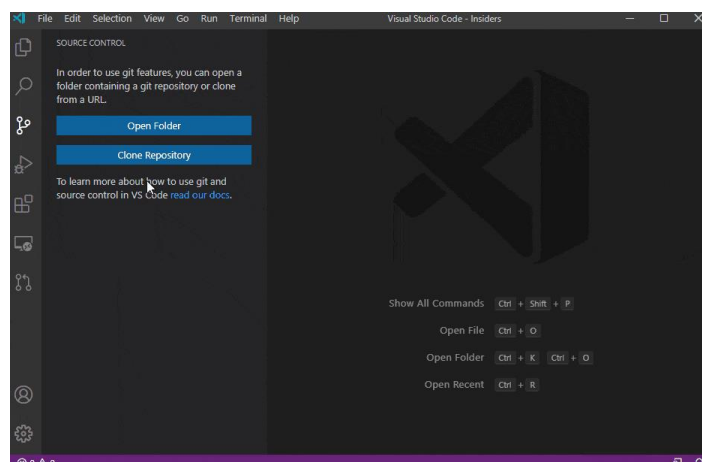


If you are not redirected to VS Code, you can add your authorization token manually. In the browser window, you will receive your authorization token. Copy the token, and switch back to VS Code. Select **Signing in to github.com...** in the Status bar, paste the token, and hit [Enter](#).

### Setting up a repository#

#### Cloning a repository#

You can search for and clone a repository from GitHub using the **Git: Clone** command in the Command Palette ([Ctrl+Shift+P](#)) or by using the **Clone Repository** button in the Source Control view (available when you have no folder open).

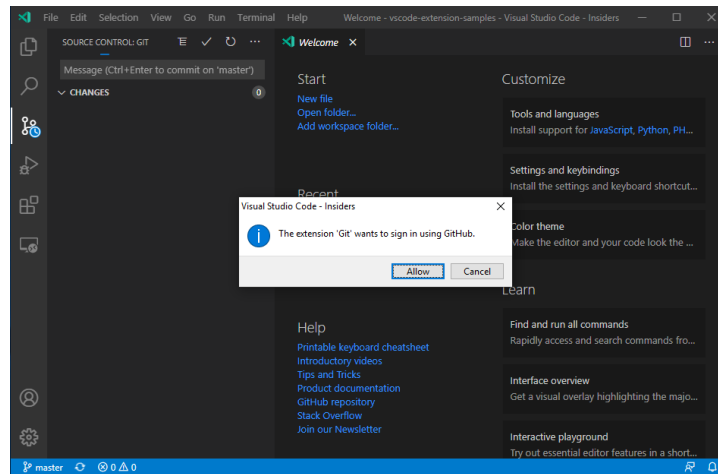


# Installation Document

## Authenticating with an existing repository#

Enabling authentication through GitHub happens when you run any Git action in VS Code that requires GitHub authentication, such as pushing to a repository that you're a member of or cloning a private repository. You don't need to have any special extensions installed for authentication; it is built into VS Code so that you can efficiently manage your repository.

When you do something that requires GitHub authentication, you'll see a prompt to sign in:



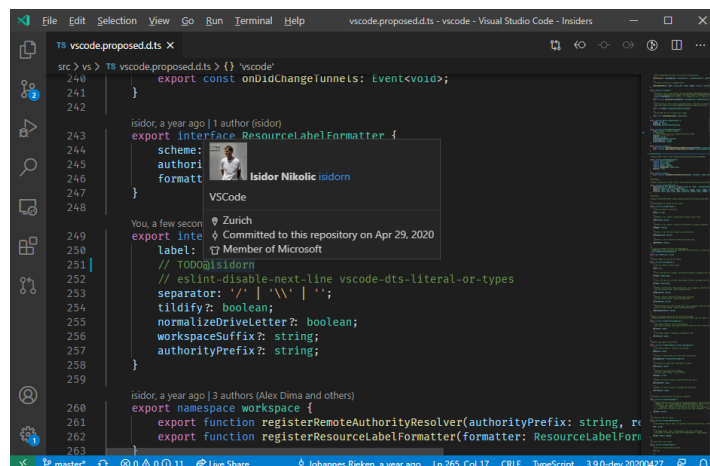
Follow the steps to sign into GitHub and return to VS Code. If authenticating with an existing repository doesn't work automatically, you may need to manually provide a personal access token. See [Personal Access Token authentication](#) for more information.

Note that there are several ways to authenticate to GitHub, including using your username and password with two-factor authentication (2FA), a personal access token, or an SSH key. See [About authentication to GitHub](#) for more information and details about each option.

## Editor integration#

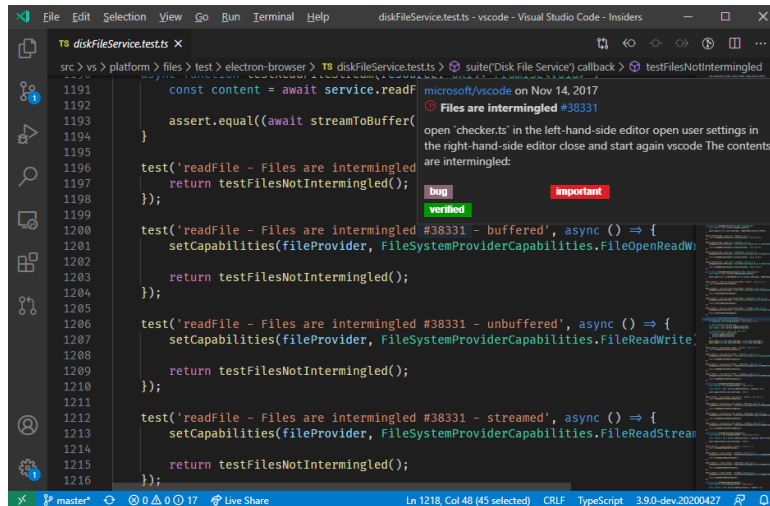
### Hovers#

When you have a repository open and a user is @-mentioned, you can hover over that username and see a GitHub-style hover.



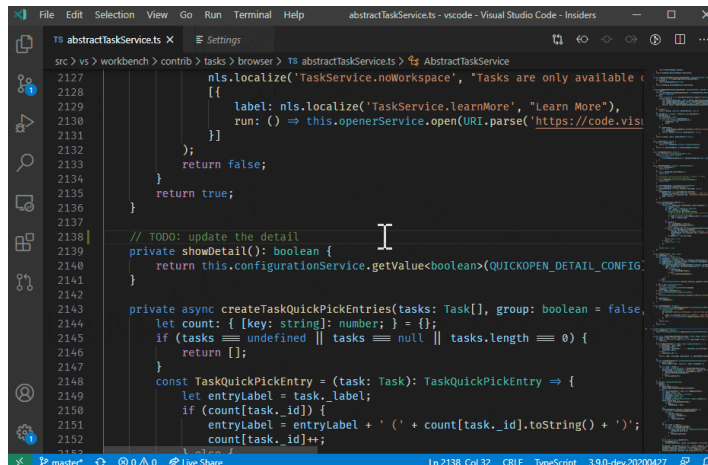
There is a similar hover for #-mentioned issue numbers, full GitHub issue URLs, and repository specified issues.

# Installation Document



## Suggestions#

User suggestions are triggered by the "@" character and issue suggestions are triggered by the "#" character. Suggestions are available in the editor and in the **Source Control** view's input box.



The issues that appear in the suggestion can be configured with the **GitHub Issues: Queries** ([githubIssues.queries](#)) [setting](#). The queries use the [GitHub search syntax](#).

You can also configure which files show these suggestions using the settings **GitHub Issues: Ignore Completion Trigger** ([githubIssues.ignoreCompletionTrigger](#)) and **GitHub Issues: Ignore User Completion Trigger** ([githubIssues.ignoreUserCompletionTrigger](#)). These settings take an array of [language identifiers](#) to specify the file types.

```
// Languages that the '#' character should not be used to trigger issue completion suggestions.

"githubIssues.ignoreCompletionTrigger": [

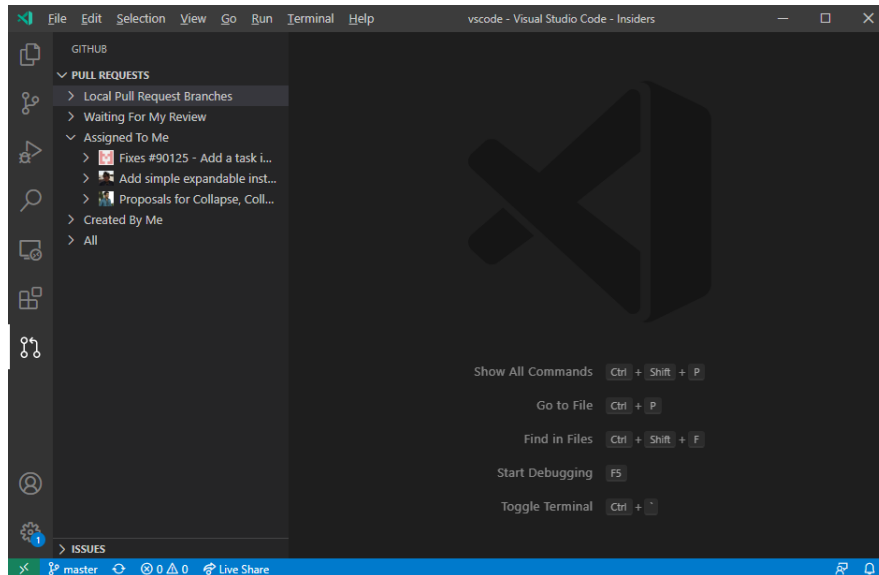
  "python"

]
```

## Pull requests#

From the **Pull Requests** view you can view, manage, and create pull requests.

# Installation Document

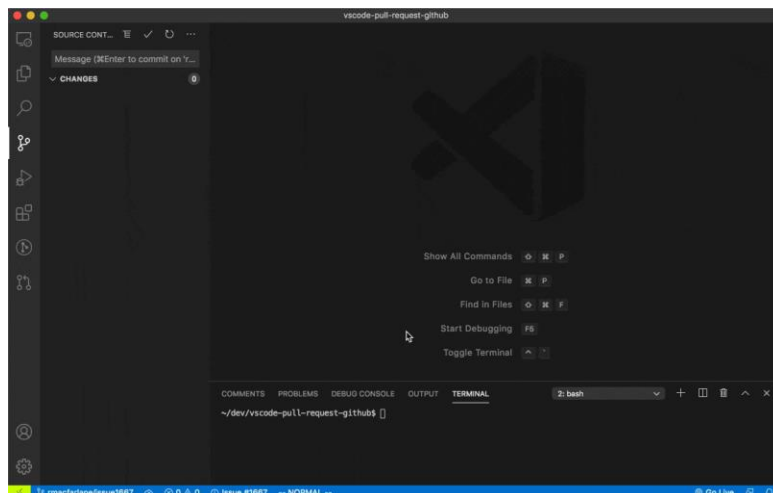


The queries used to display pull requests can be configured with the **GitHub Pull Requests: Queries** (`githubPullRequests.queries`) setting and use the [GitHub search syntax](#).

```
"githubPullRequests.queries": [  
  
  {  
  
    "label": "Assigned To Me",  
  
    "query": "is:open assignee:${user}"  
  
  },  
]
```

## Creating Pull Requests#

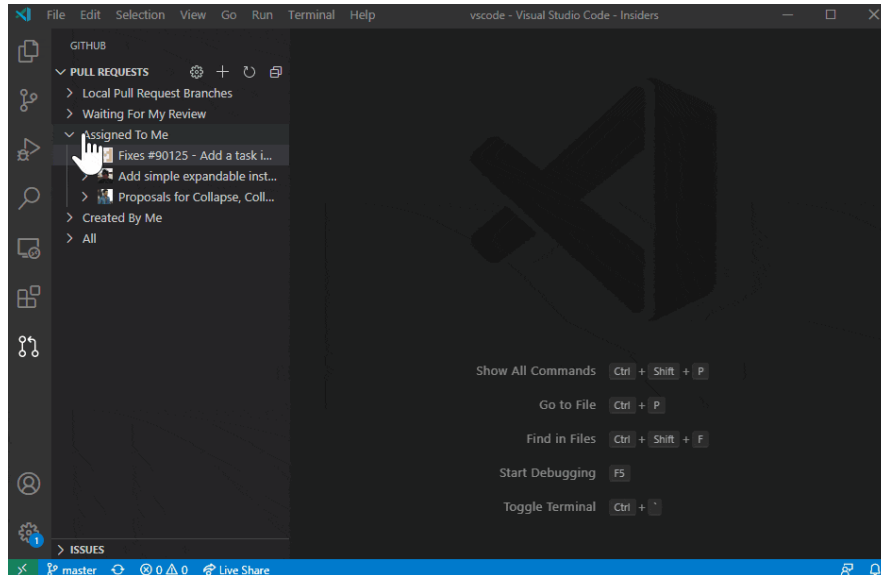
You can use the **GitHub Pull Requests: Create Pull Request** command or use the + button in the **Pull Requests** view to create a pull request. If you have not already pushed your branch to a remote, the extension will do this for you. You can use the last commit message, the branch name, or write a custom title for the pull request. If your repository has a pull request template, this will automatically be used for the description.



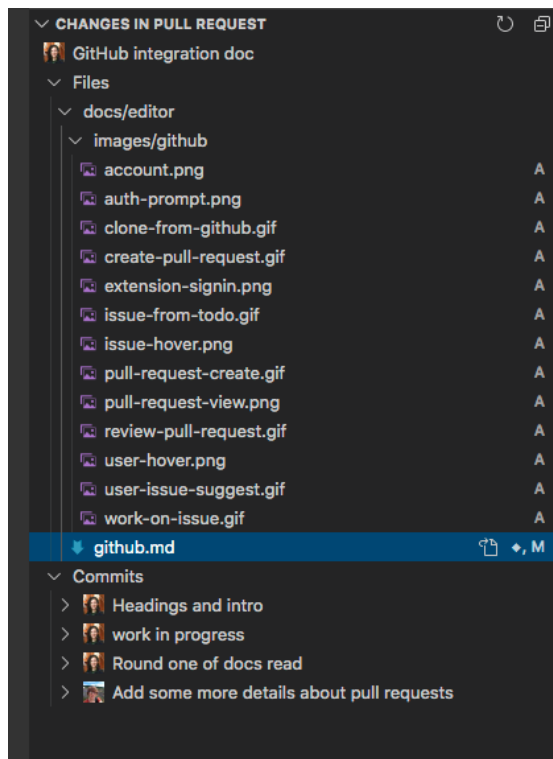
# Installation Document

## Reviewing#

Pull requests can be reviewed from the **Pull Requests** view. You can assign reviewers and labels, add comments, approve, close, and merge all from the pull request description.



From the description page, you can also easily checkout the pull request locally using the **Checkout** button. This will add a new **Changes in Pull Request** view from which you can view diffs of the current changes as well as all commits and the changes within these commits. Files that have been commented on are decorated with a diamond icon. To view the file on disk, you can use the **Open File** inline action.



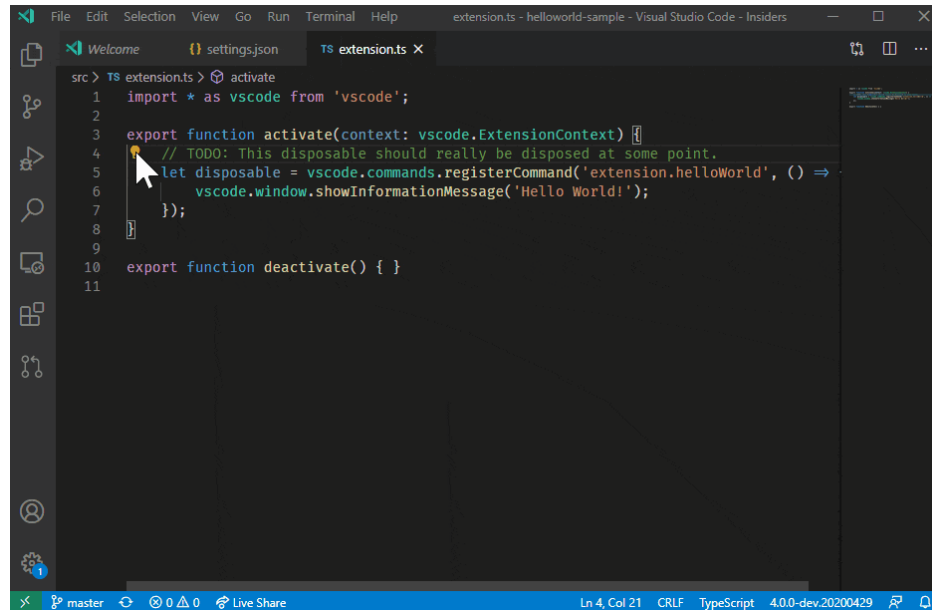
The diff editors from this view use the local file, so file navigation, IntelliSense, and editing work as normal. You can add comments within the editor on these diffs. Both adding single comments and creating a whole review is supported.

# Installation Document

## Issues#

### Creating issues#

Issues can be created from the + button in the **Issues** view and by using the **GitHub Issues: Create Issue from Selection** and **GitHub Issues: Create Issue from Clipboard** commands. They can also be created using a Code Action for "TODO" comments.



You can configure the trigger for the Code Action using the **GitHub Issues: Create Issue Triggers** (`githubIssues.createIssueTriggers`) setting.

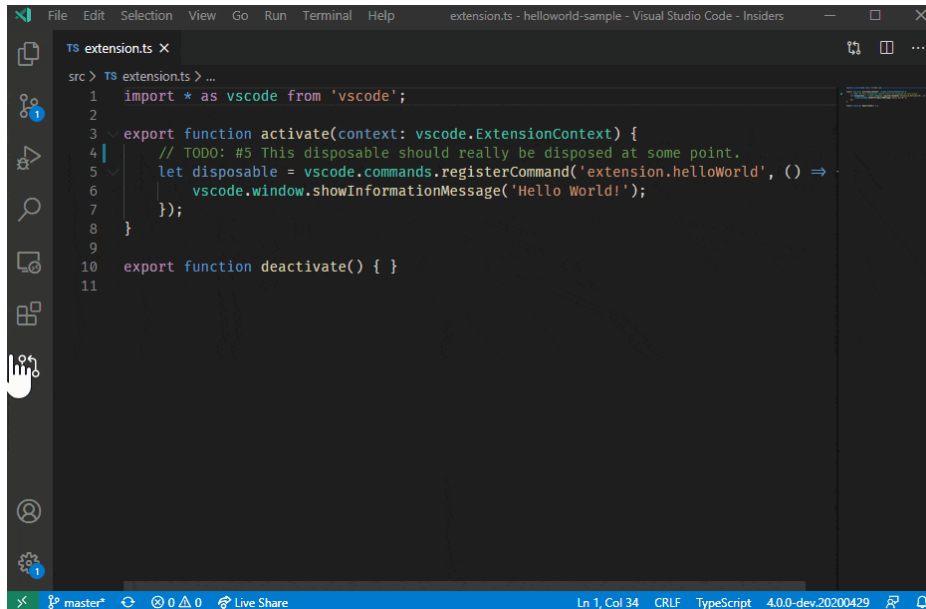
The default issue triggers are:

```
"githubIssues.createIssueTriggers": [  
  "TODO",  
  "todo",  
  "BUG",  
  "FIXME",  
  "ISSUE",  
  "HACK"  
]
```

### Working on issues#

From the **Issues** view, you can see your issues and work on them. By default, when you start working on an issue, a branch will be created for you. You can configure the name of the branch using the **GitHub Issues: Working Issue Branch** (`githubIssues.workingIssueBranch`) setting. The commit message input box in the **Source Control** view will be populated with a commit message, which can be configured with **GitHub Issues: Working Issue Format SCM** (`githubIssues.workingIssueFormatScm`).

# Installation Document



```
src > TS extension.ts > ...
1  import * as vscode from 'vscode';
2
3  export function activate(context: vscode.ExtensionContext) {
4      // TODO: #5 This disposable should really be disposed at some point.
5      let disposable = vscode.commands.registerCommand('extension.helloWorld', () => {
6          vscode.window.showInformationMessage('Hello World!');
7      });
8  }
9
10 export function deactivate() { }
11
```

If your workflow doesn't involve creating a branch, or if you want to be prompted to enter a branch name every time, you can skip that step by turning off the **GitHub Issues: Use Branch For Issues** ([githubIssues.useBranchForIssues](#)) setting.



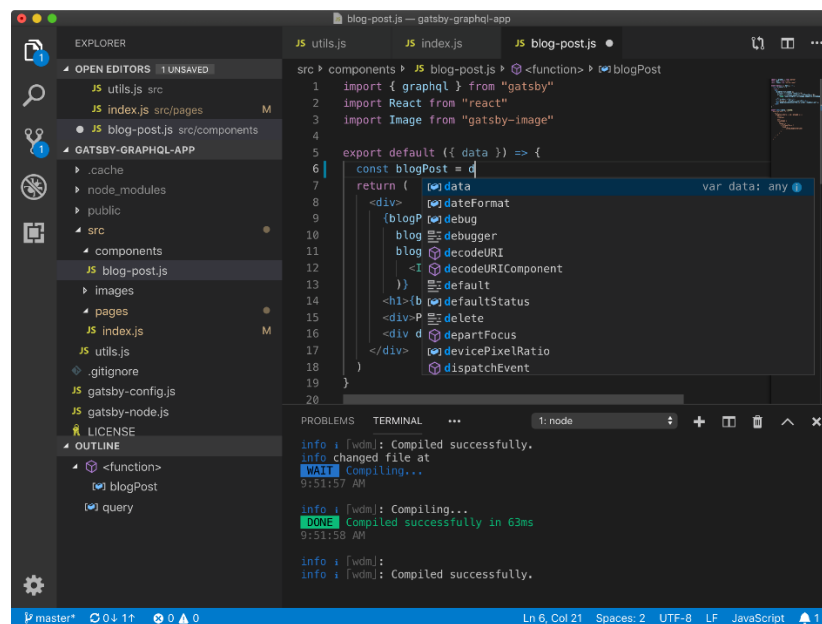
# Installation Document

## Visual Studio Code - Open Source ("Code - OSS")

### The Repository

This repository ("Code - OSS") is where we (Microsoft) develop the [Visual Studio Code](#) product. Not only do we work on code and issues here, we also publish our [roadmap](#), [monthly iteration plans](#), and our [endgame plans](#). This source code is available to everyone under the standard [MIT license](#).

### Visual Studio Code



[Visual Studio Code](#) is a distribution of the Code - OSS repository with Microsoft specific customizations released under a traditional [Microsoft product license](#).

[Visual Studio Code](#) combines the simplicity of a code editor with what developers need for their core edit-build-debug cycle. It provides comprehensive code editing, navigation, and understanding support along with lightweight debugging, a rich extensibility model, and lightweight integration with existing tools.

Visual Studio Code is updated monthly with new features and bug fixes. You can download it for Windows, macOS, and Linux on [Visual Studio Code's website](#). To get the latest releases every day, install the [Insiders build](#).

### Contributing

There are many ways in which you can participate in the project, for example:

# Installation Document

- [Submit bugs and feature requests](#), and help us verify as they are checked in
- Review [source code changes](#)
- Review the [documentation](#) and make pull requests for anything from typos to new content

If you are interested in fixing issues and contributing directly to the code base, please see the document [How to Contribute](#), which covers the following:

- [How to build and run from source](#)
- [The development workflow, including debugging and running tests](#)
- [Coding guidelines](#)
- [Submitting pull requests](#)
- [Finding an issue to work on](#)
- [Contributing to translations](#)

## Feedback

- Ask a question on [Stack Overflow](#)
- [Request a new feature](#)
- Upvote [popular feature requests](#)
- [File an issue](#)
- Follow [@code](#) and let us know what you think!

See our [wiki](#) for a description of each of these channels and information on some other available community-driven channels.

## Related Projects

Many of the core components and extensions to VS Code live in their own repositories on GitHub. For example, the [node debug adapter](#) and the [mono debug adapter](#) have their own repositories. For a complete list, please visit the [Related Projects](#) page on our [wiki](#).

## Bundled Extensions

VS Code includes a set of built-in extensions located in the [extensions](#) folder, including grammars and snippets for many languages. Extensions that provide rich language support (code completion, Go to Definition) for a language have the suffix language-features. For example, the json extension provides coloring for JSON and the json-language-features provides rich language support for JSON.

## Development Container

This repository includes a Visual Studio Code Remote - Containers / Codespaces development container.

- For [Remote - Containers](#), use the **Remote-Containers: Open Repository in Container...** command which creates a Docker volume for better disk I/O on macOS and Windows.
- For Codespaces, install the [Visual Studio Codespaces](#) extension in VS Code, and use the **Codespaces: Create New Codespace** command.

# Installation Document

Docker / the Codespace should have at least **4 Cores and 6 GB of RAM (8 GB recommended)** to run full build. See the [development container README](#) for more information.

## Code of Conduct

This project has adopted the [Microsoft Open Source Code of Conduct](#). For more information see the [Code of Conduct FAQ](#) or contact [opencode@microsoft.com](mailto:opencode@microsoft.com) with any additional questions or comments.

## License

Copyright (c) Microsoft Corporation. All rights reserved.

Licensed under the [MIT](#) license.