Git

Ignore the Env Files in Git

7 months ago • by John Otieno

Whether you are just getting started with Git or a seasoned pro, one of the most fundamental features you will use is the “gitignore”.

It is common to have files and directories that contain sensitive information such as API keys, passwords, configuration files, etc. It is therefore dangerous to include such files into a public or even private repository where other users can find and use them.

Luckily, Git has the “gitignore” feature that allows us to tell Git not to include these files and directories into the repository.

In this tutorial, we will explore the fundamentals of the “gitignore” feature and how we can use it to exclude specific files and directories from being tracked by Git. We will then focus on how to use the “gitignore” feature to ignore the environment variables and other sensitive files.

Gitignore

In Git, gitignore is a fundamental feature that allows us to configure the files and directories that are excluded from version control tracking.

It is simply a text file containing entries for the names of directories and files that we wish to exclude. We can also include the patterns in the file to determine the files that should be excluded.

Creating a Gitignore File

Before we dive into the process of configuring the “gitignore” file, we need to ensure that we have the file present in the target repository.

Start by navigating to the root directory of the Git repository. For example, suppose we have a repository called “hello_world”.

We can create a file by navigating to the following directory:

```
$ cd ~/src/hello_world
```

In the root directory, create a new file called ".gitignore". This is where we store all the entries for all files and directories that we wish the version control system to ignore.

```
$ touch .gitignore
```

Adding Entries to the Gitignore Files

Once we created the "gitignore" file, we can proceed and add the files and directories that we wish to include.

You can do this by opening the "gitignore" file with a text editor. Next, add the patterns to specify the files and directories to ignore.

Each pattern should be on a separate line. For example, suppose we are working within a Python-based repository and we exclude the Python compiled bytecode and virtual environment directories.

We can add the entries as follows:

```
__pycache__/
venv/
```

This should ensure that Git does not track any "pycache" files and any file that is stored within the "venv" directory.

Ignore the Env Files

In the world of development, you will find most of the developers who tend to use ".env" files to store the environment variables and sensitive information.

To tell Git to ignore any ".env" files, we can add the entry in the "gitignore" file as shown in the following example:

Start by editing the ".gitignore" file and add a pattern to ignore the ".env" files.

```
*.env
```

This should force Git to ignore all files that end in ".env" extension.

We can use a wildcard (*) to match all ".env" files as shown previously or specify the exact filename. For example, to exclude a file called "terraform.tf.env", we can add the entry as follows:

```
terraform.tf.env
```

Removing the Already Included Files