Setting up a GIT repo for Django Application using PyCharm

SummitWorks™
GLOBAL SOLUTION ARCHITECTS

Authorized & published by Summitworks Technologies Inc

Agenda

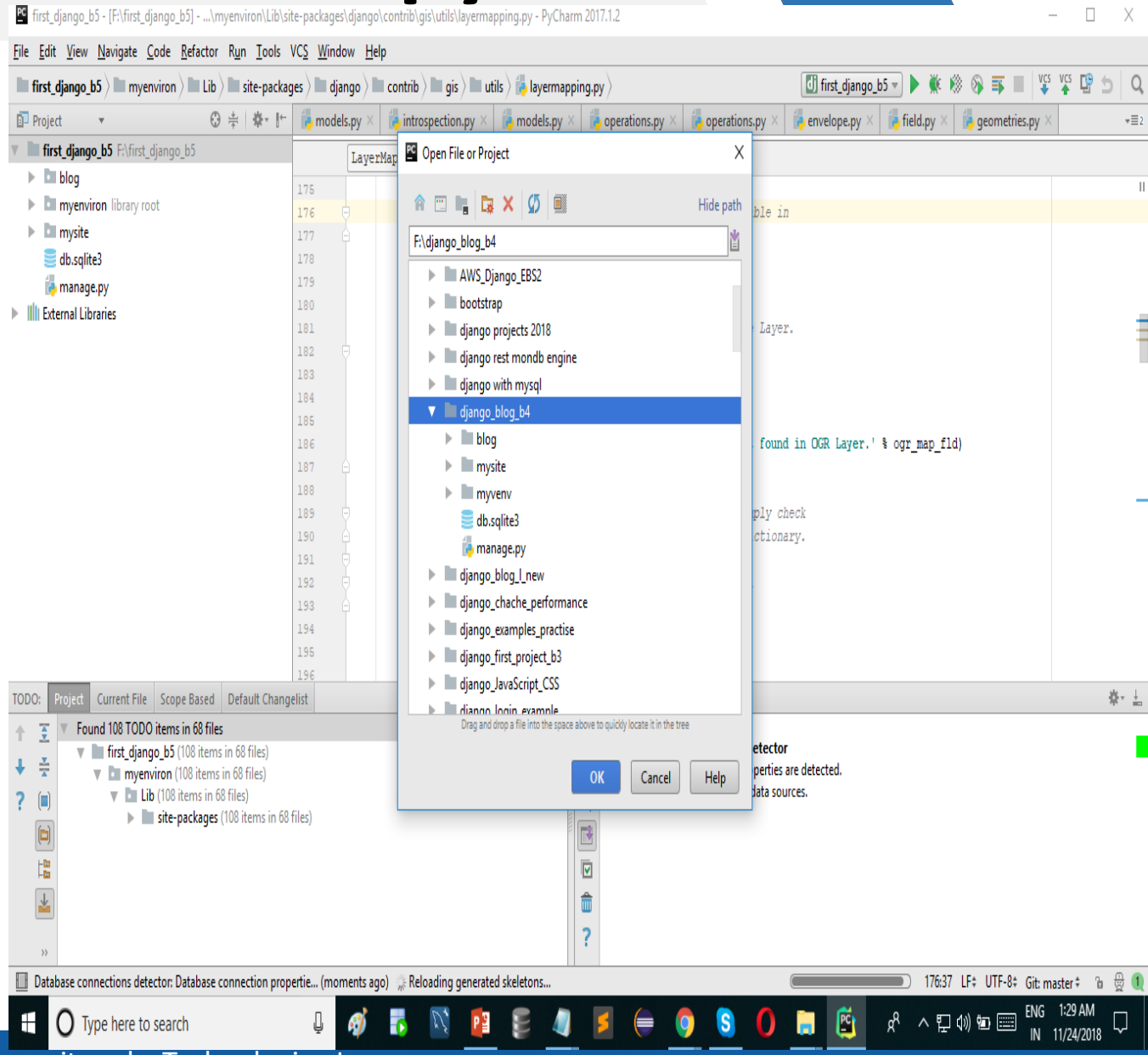
- **Set up a Git repository for Django Application**

Git

- **Git** is a revision control system, a tool to manage your source code history.
- Its for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files.
- **You can use IDE tools or Commands to work with GIT**

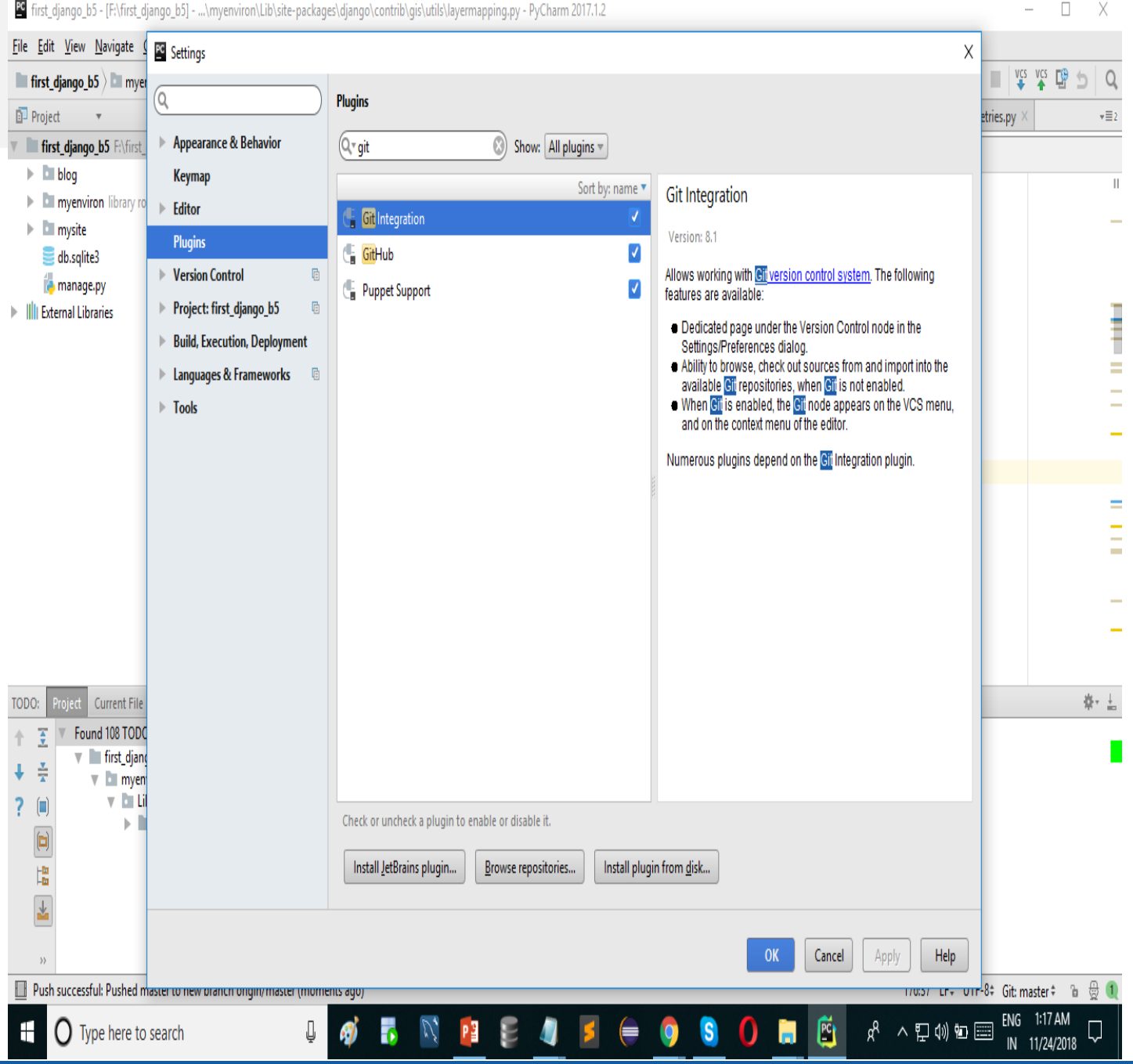
Open virtual environment into pycharm

Use pycharm professional edition.[its trail version]
Open virtual environment or create a new Django project in pycharm.
You can open virtual environment like below.
Click file→ open project → select the location of virtual environment.
[make sure you configure python interpreter before you create / import Django project.]



Git in Pycharm

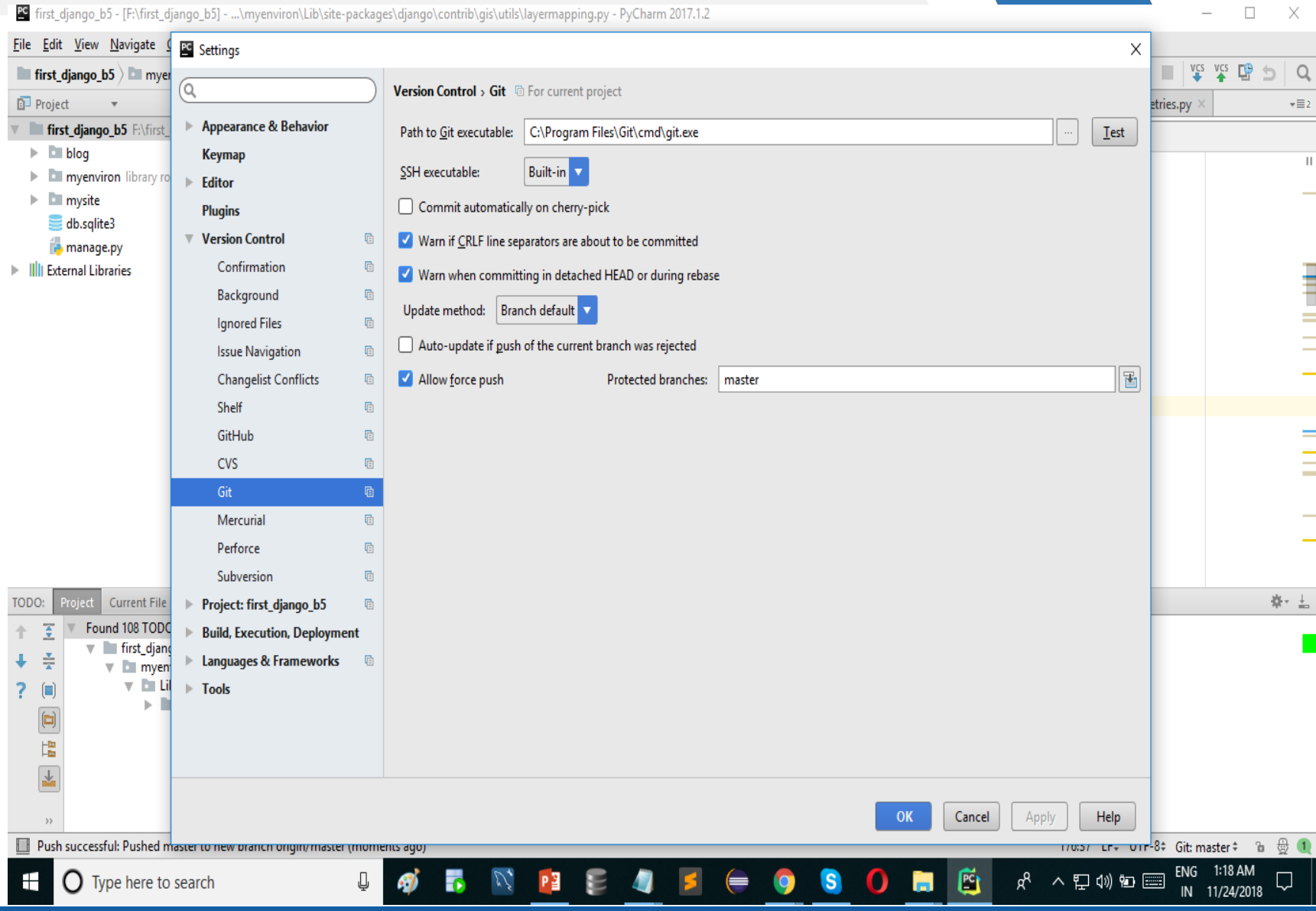
To be able to use Git for version control, make sure that the Git Integration plugin is enabled in the Settings/Preferences dialog (Ctrl+Alt+S) under Plugins.



Git in Pycharm

Before you can enable Git version control for an existing local project, or clone a Git project from a remote repository, do the following:

- Download and install Git.
- In the Settings/Preferences dialog (Ctrl+Alt+S), select Version Control | Git in the left pane and specify the path to the Git executable.



Set up a Git repository - pycharm

- **Check out a project from a remote host (clone):** PyCharm allows you to check out (in Git terms **clone**) an existing repository and create a new project based on the data you've downloaded.

  - From the main menu, choose VCS | Checkout from Version Control | Git, or, if no project is currently opened, choose Checkout from Version Control | Git on the Welcome screen.

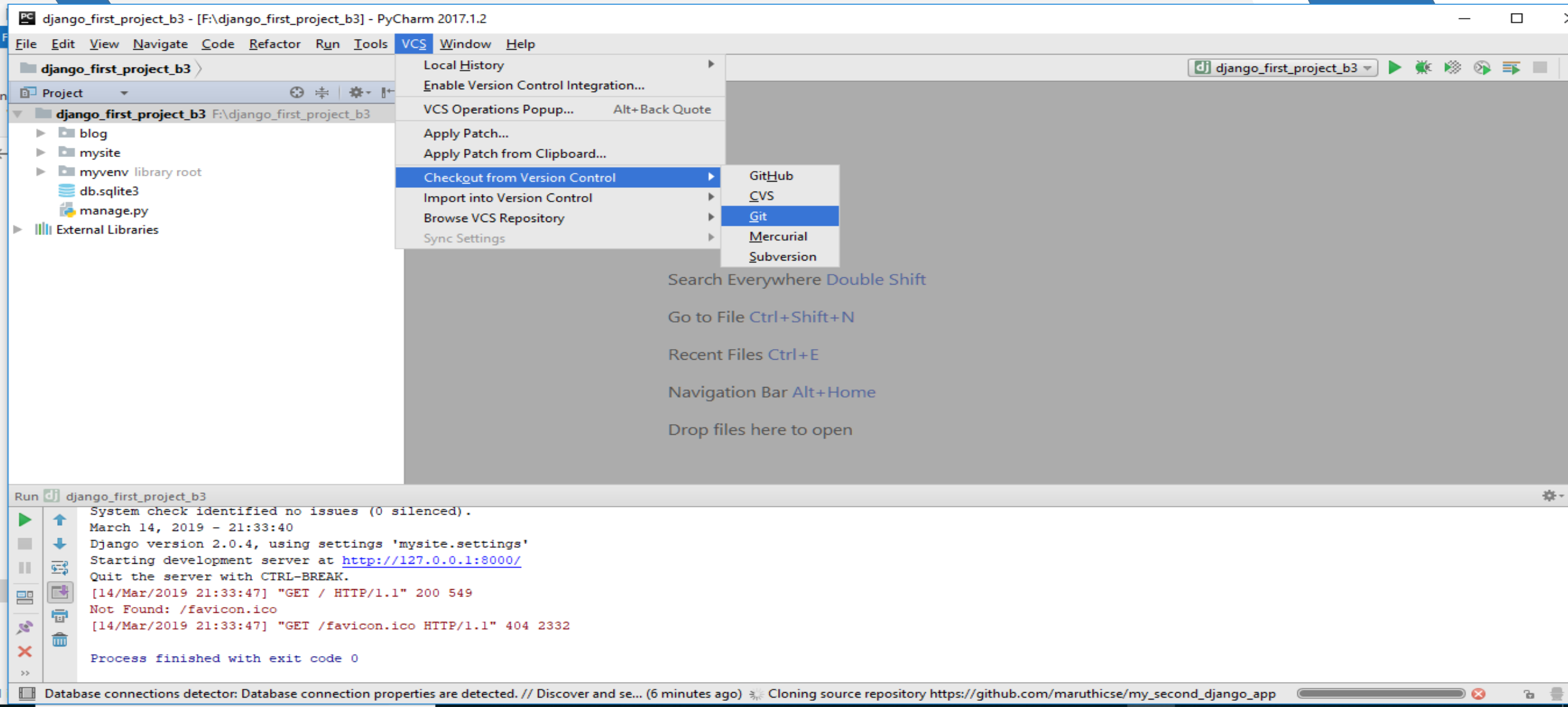
  - In the Clone Repository dialog, specify the URL of the remote repository you want to clone (you can click Test to make sure that connection to the remote can be established).

  - In the Directory field, specify the path where the folder for your local Git repository will be created into which the remote repository will be cloned.

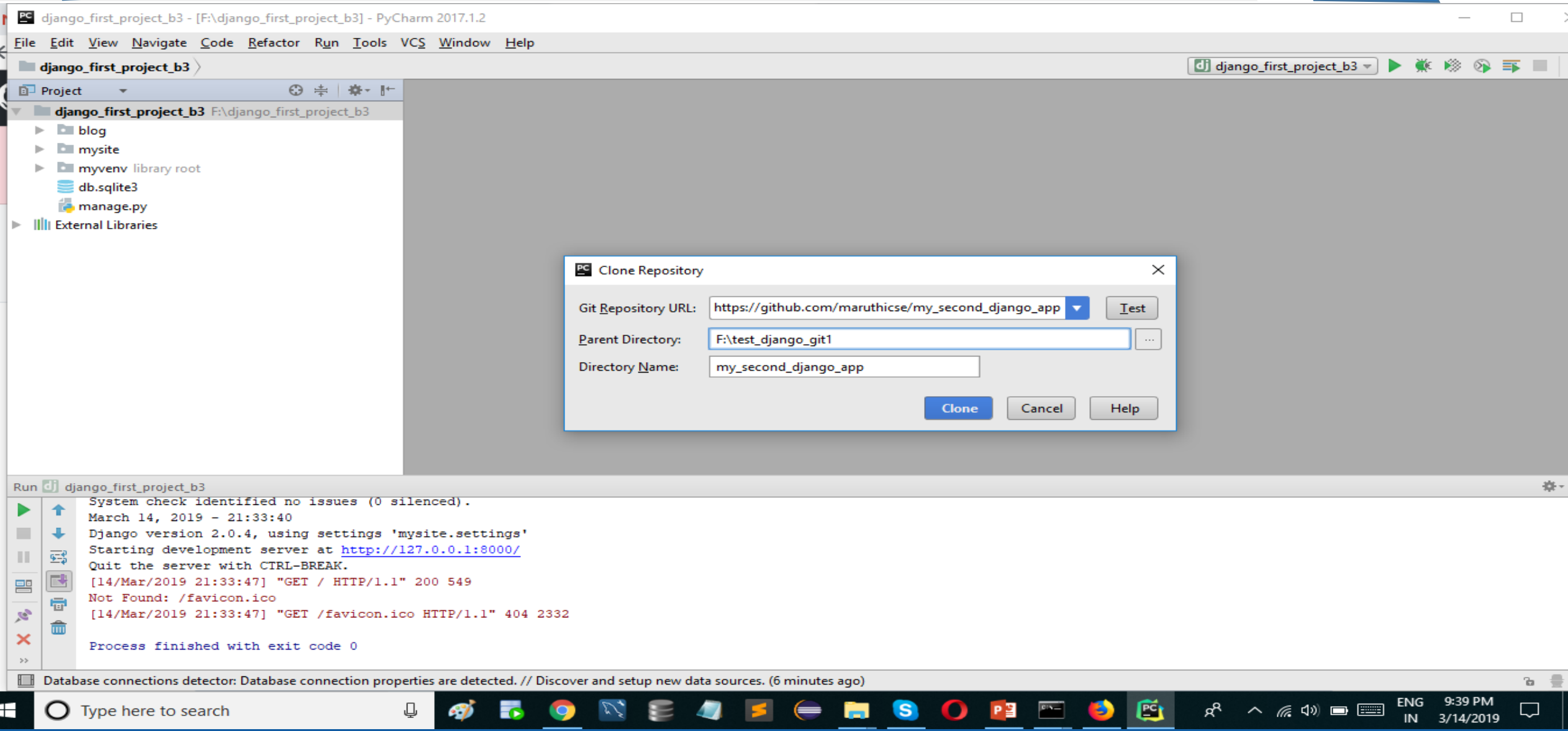
  - Click Clone. If you want to create a PyCharm project based on the sources you have cloned, click Yes in the confirmation dialog. Git root mapping will be automatically set to the project root directory.

If your project contains submodules, they will also be cloned and automatically registered as project roots.

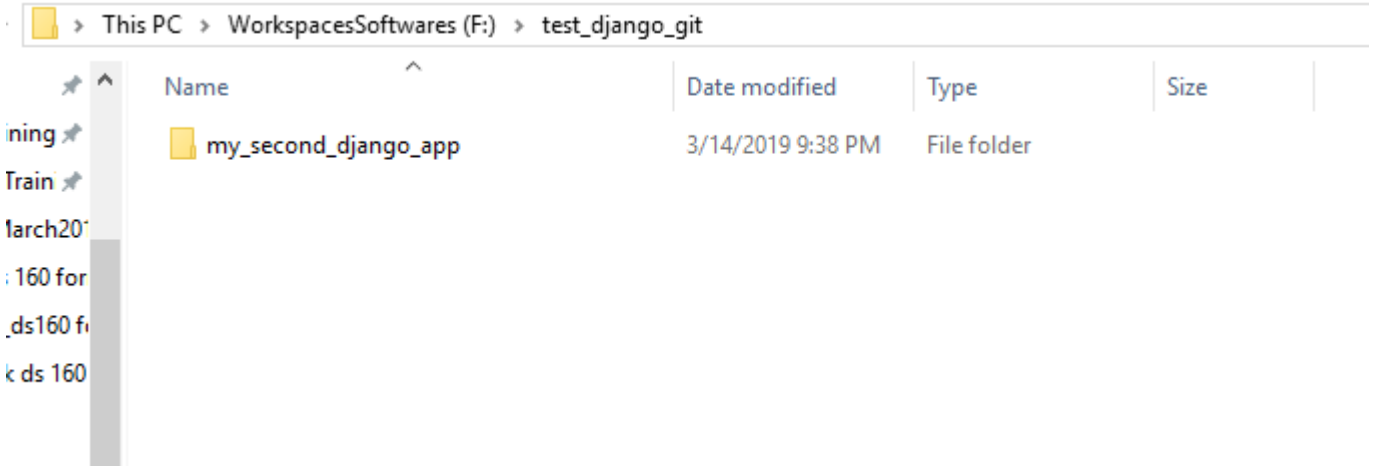
Set up a Git repository - pycharm



Set up a Git repository - pycharm



Set up a Git repository - pycharm



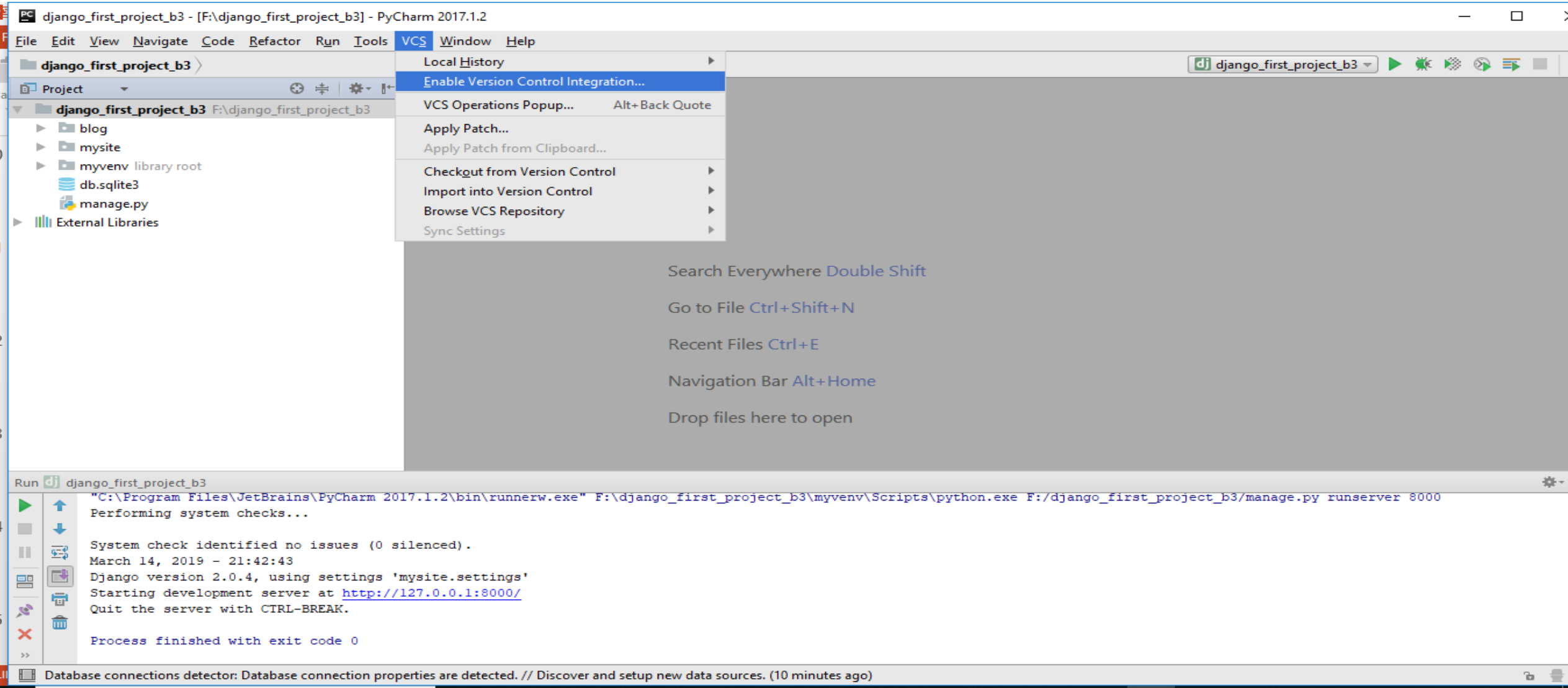
Put an existing project under Git version control

Apart from cloning a remote repository, you can create a local repository based on an existing project's sources.

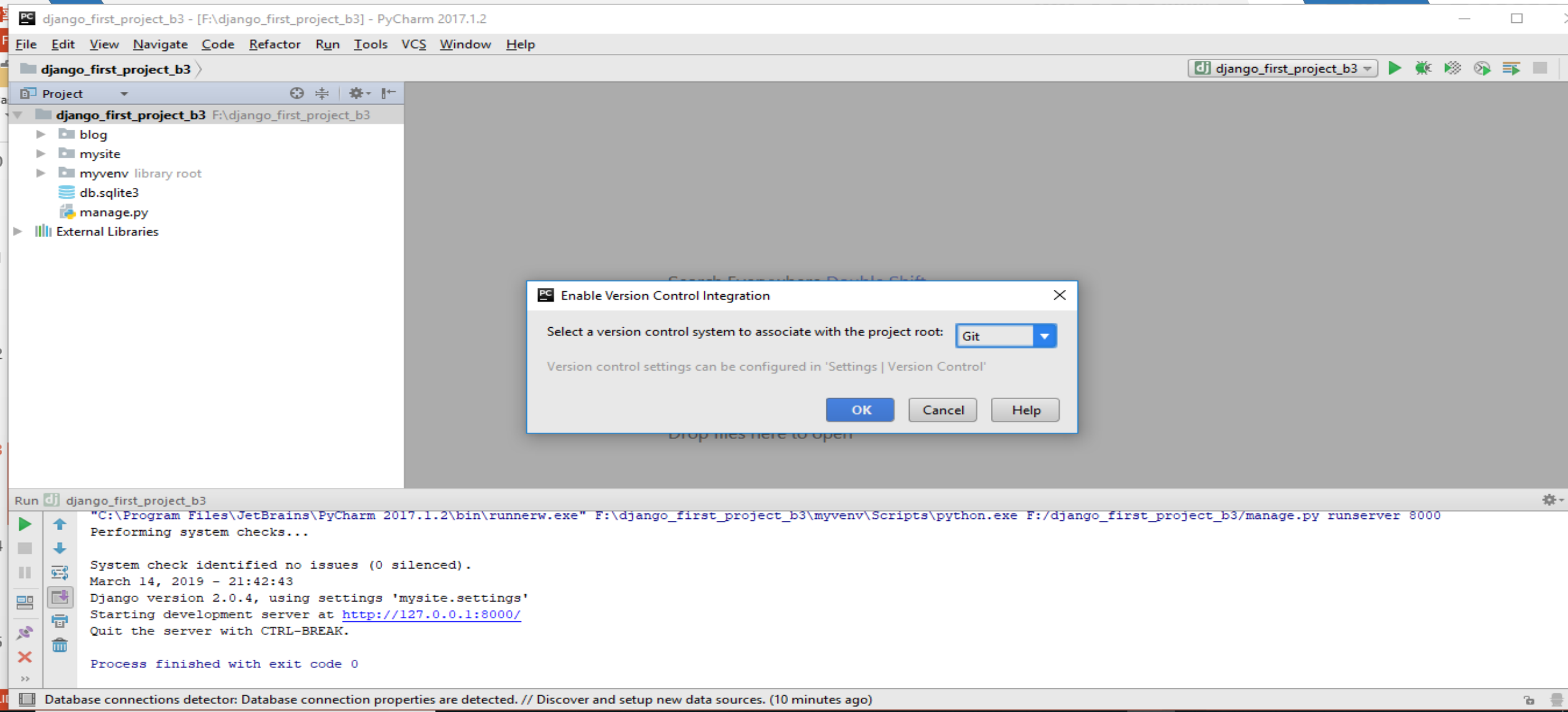
Import an entire project into a single Git repository

- Open the project that you want to put under Git.
- From the main menu, choose VCS | Enable Version Control Integration.
- In the dialog that opens, select Git from the drop-down list and click OK.

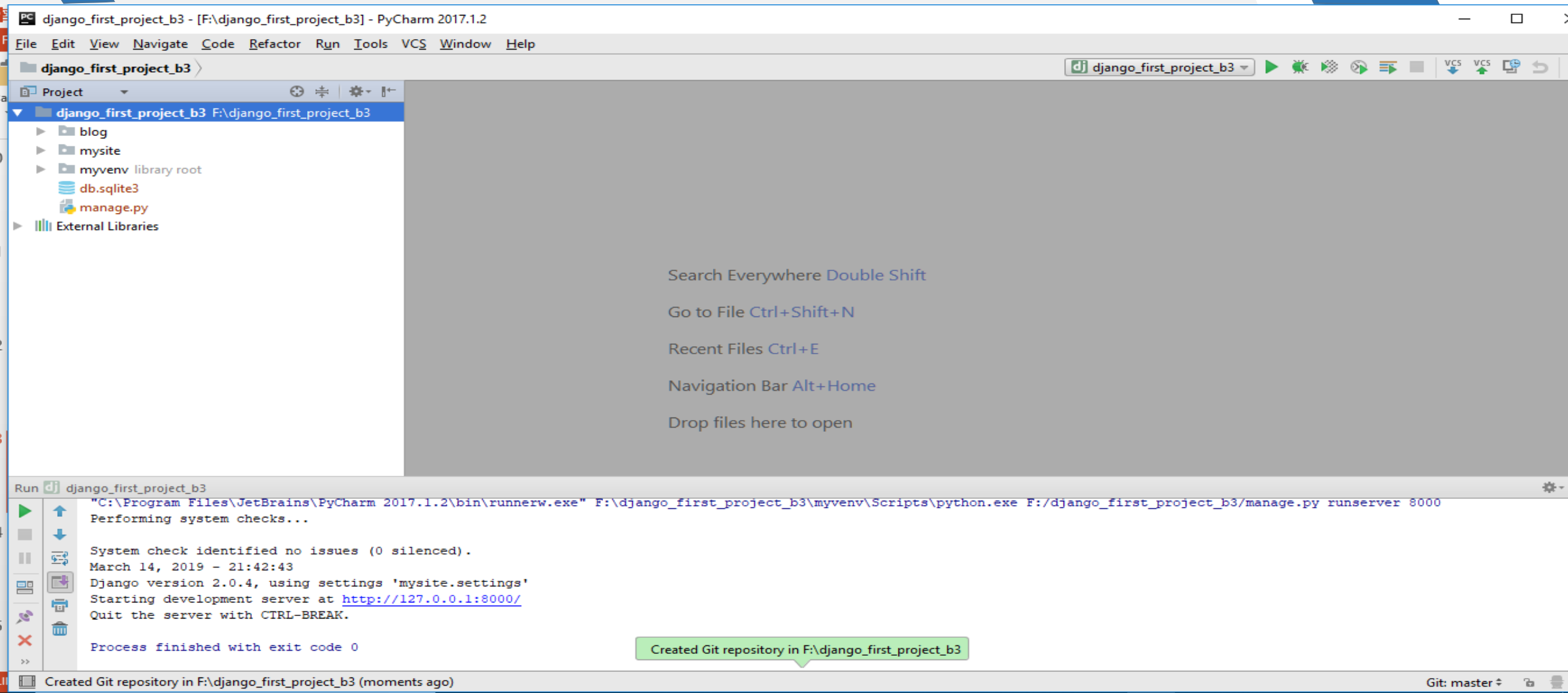
Put an existing project under Git version control



Put an existing project under Git version control



Put an existing project under Git version control



Add files to the local repository

After you have initialized a Git repository for your project, you need to add project data to it.

Open the Version Control tool window (Alt+9) and switch to the Local Changes tab.

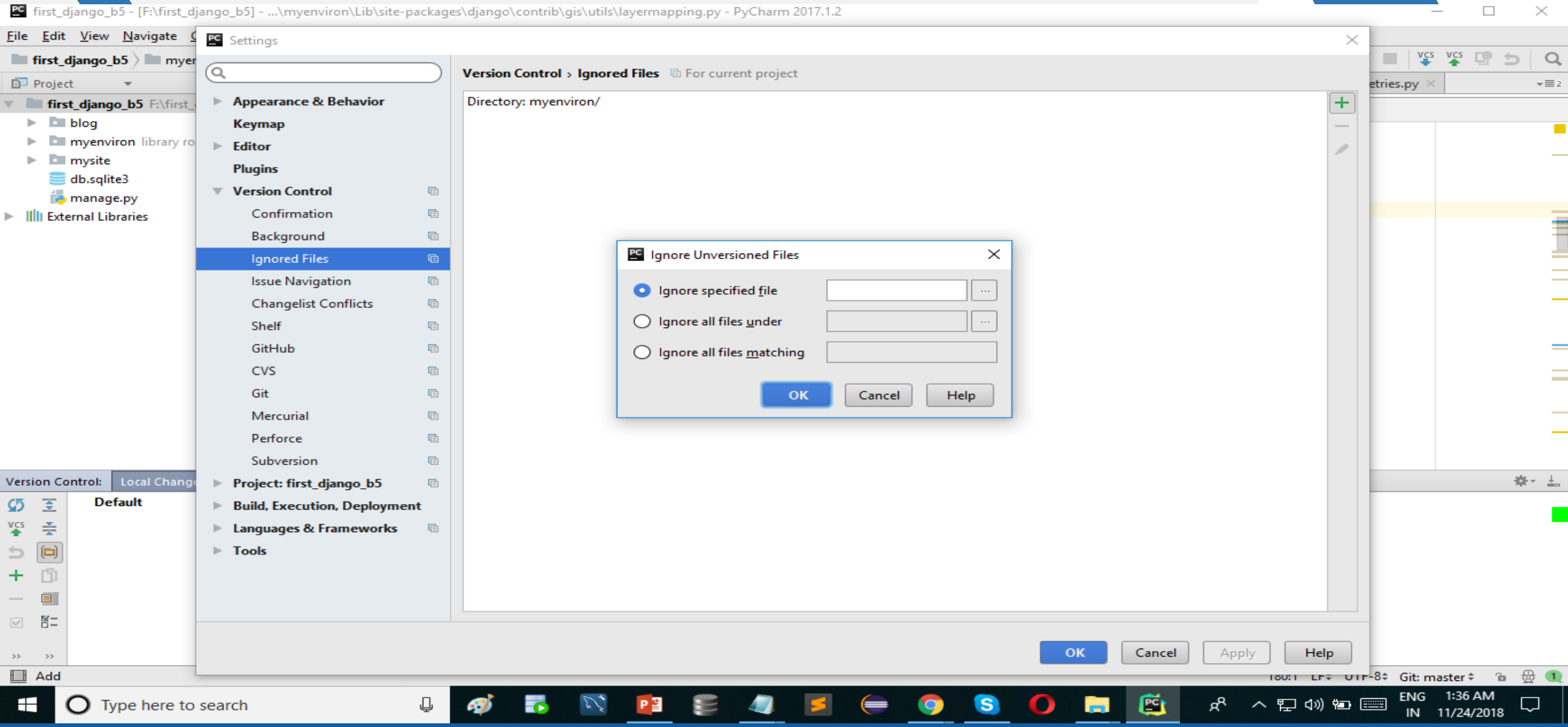
Put any files in the Unversioned Files changelist under version control by pressing Ctrl+Alt+A or selecting Add to VCS from the context menu. You can either add the entire changelist, or select separate files.

If you have enabled Git integration for your project, PyCharm suggests to add each newly created file under Git version control (you can change this behavior in the Settings dialog (Ctrl+Alt+S) under Version Control | Confirmation). If you want certain files to always remain unversioned, you can configure Git to ignore them.

Exclude files from version control (ignore)

- **Configure a list of files to be ignored by Git**
- Either:
  - In the Settings/Preferences dialog (Ctrl+Alt+S) select Version Control | Ignored Files in the left pane.
  - Open the Version Control tool window (Alt+9) and switch to the Local Changes tab. Click eye icon on the toolbar and choose Configure Ignored Files.
- Click the Add add button on the toolbar.
- In the Ignore Unversioned Files dialog, specify the files/directories that you want to ignore, or define file name patterns:
  - Ignore specified file: specify the file name relative to the project root.
  - Ignore all files under: specify the directory whose contents should be ignored relative to the project root. The rule is applied recursively to all subdirectories.
  - Ignore all files matching: type the pattern that defines the names of files to be ignored. The rule is applied to all directories under the project root.
  - Two characters can be used as wildcards:
  - *: to replace any string.
  - ?: to replace a single character.
- For example, *.iml will ignore all files with the iml extension; *.?ml will ignore all files whose extension ends with ml.

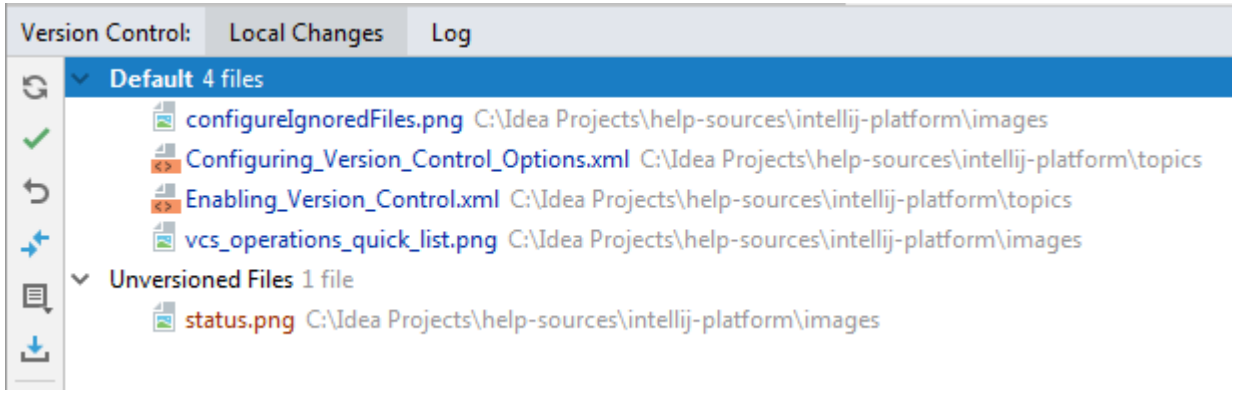
Exclude files from version control (ignore)



Check project status

PyCharm allows you to check the status of your local working copy compared to the repository version of the project. It lets you see which files have been modified, which new files have been added to the VCS, and which files are not being tracked by Git.

Open the Version Control tool window (Alt+9) and switch to the Local Changes tab:



The Default changelist shows all files that have been modified since you last synchronized with the remote repository (highlighted in blue), and all new files that have been added to the VCS but have not been committed yet (highlighted in green).

The Unversioned Files changelist shows all files that have been added to your project, but that are not being tracked by Git.

https://www.jetbrains.com/help/pycharm/set-up-a-git-repository.html#put-existing-project-under-Git

Add a remote repository

To be able to collaborate on your Git project, you need to configure remote repositories that you fetch data from and push to when you need to share your work.

If you have cloned a remote Git repository, for example from GitHub, the remote is configured automatically and you do not have to specify it when you want to synchronize with it (i.e. when you perform a pull or a push operation).

The default name Git gives to the remote you've cloned from is **origin**.

However, if you created a Git repository based on local sources, you need to add a remote repository for other contributors to be able to push their changes to it, and for you to be able to share the results of your work.

https://www.jetbrains.com/help/pycharm/set-up-a-git-repository.html#put-existing-project-under-Git

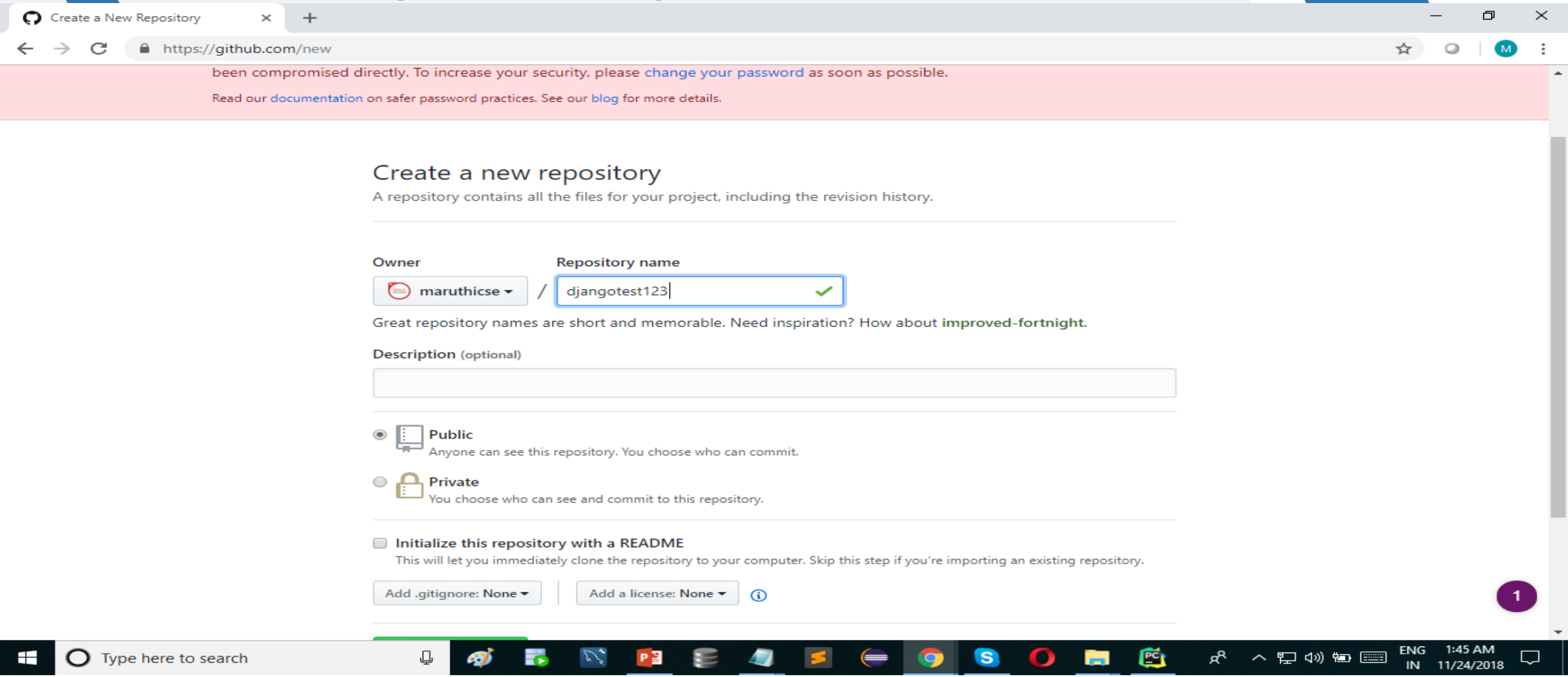
Adding remotes

Before this, create URL in git hub.

1. From the main menu, choose VCS | Git | Remotes. The Git Remotes dialog will open.
2. Click the Add add button on the toolbar or press Alt+Insert.
3. In the dialog that opens, specify the remote name and URL and click OK.

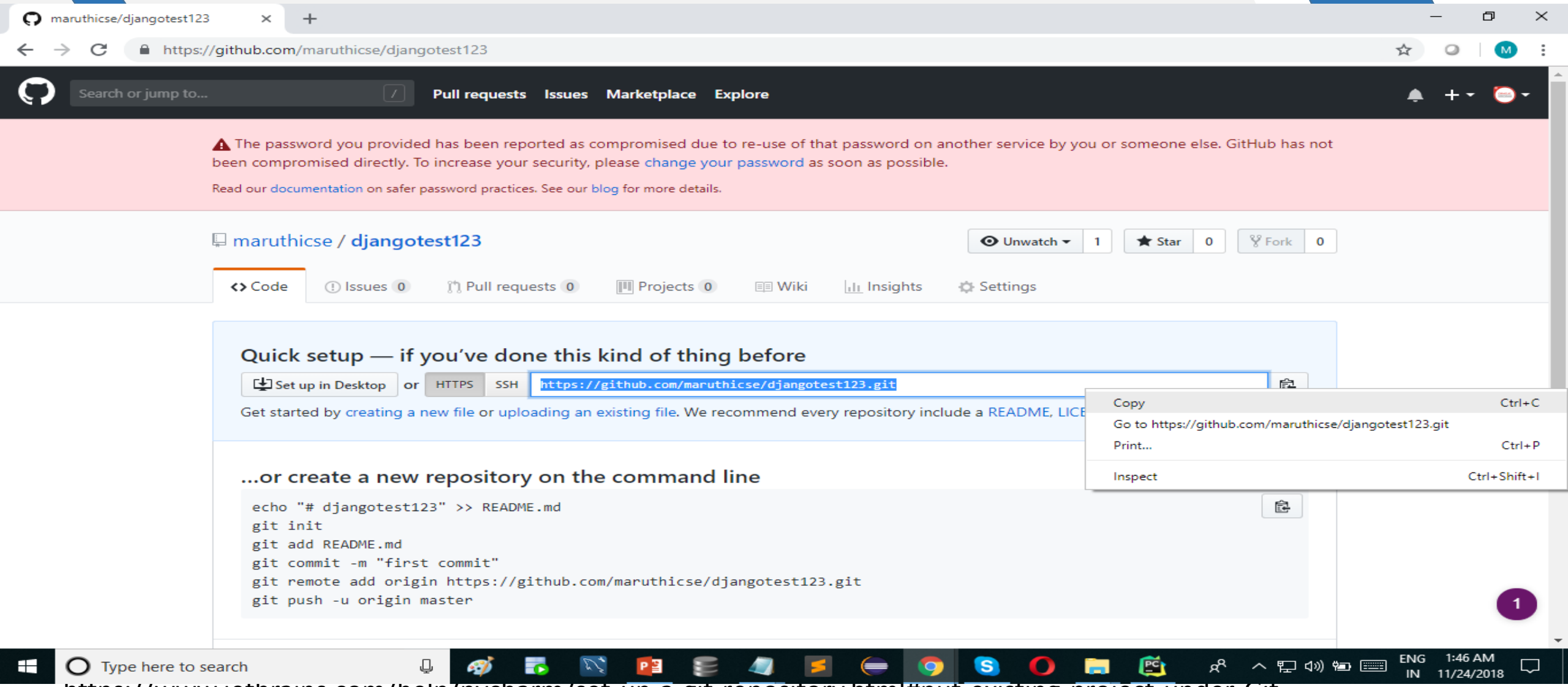
https://www.jetbrains.com/help/pycharm/set-up-a-git-repository.html#put-existing-project-under-Git

Creating URL in git



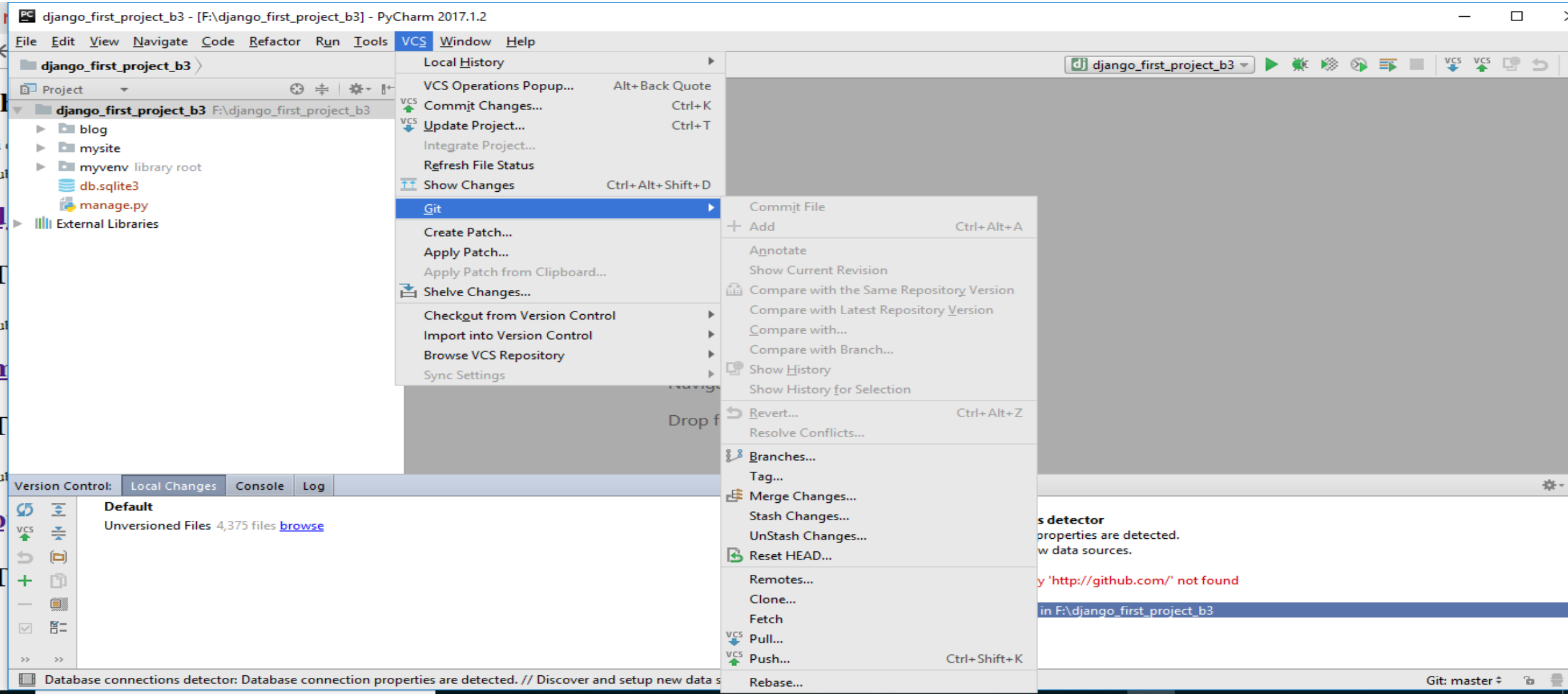
https://www.jetbrains.com/help/pycharm/set-up-a-git-repository.html#put-existing-project-under-Git

Creating URL in git

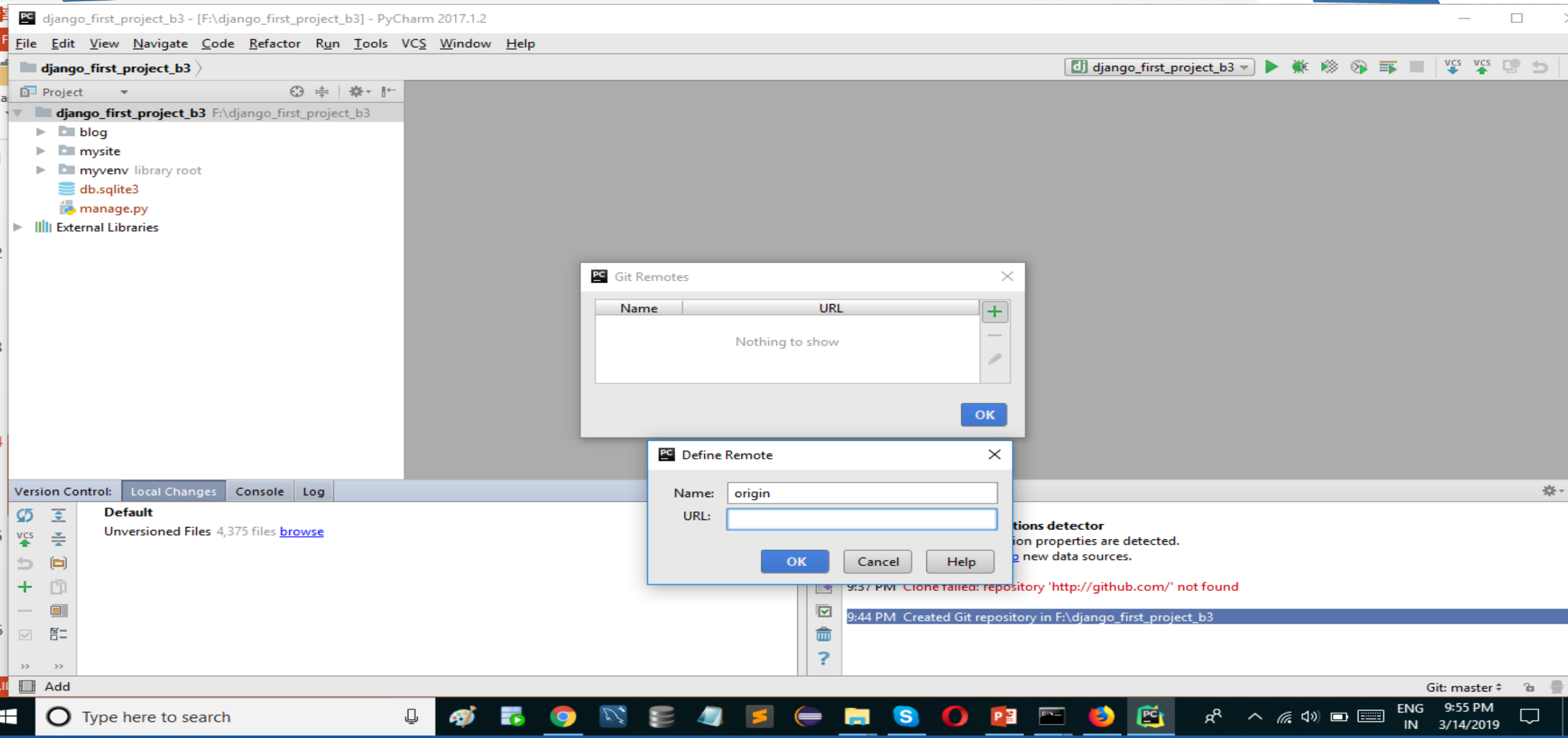


https://www.jetbrains.com/help/pycharm/set-up-a-git-repository.html#put-existing-project-under-Git

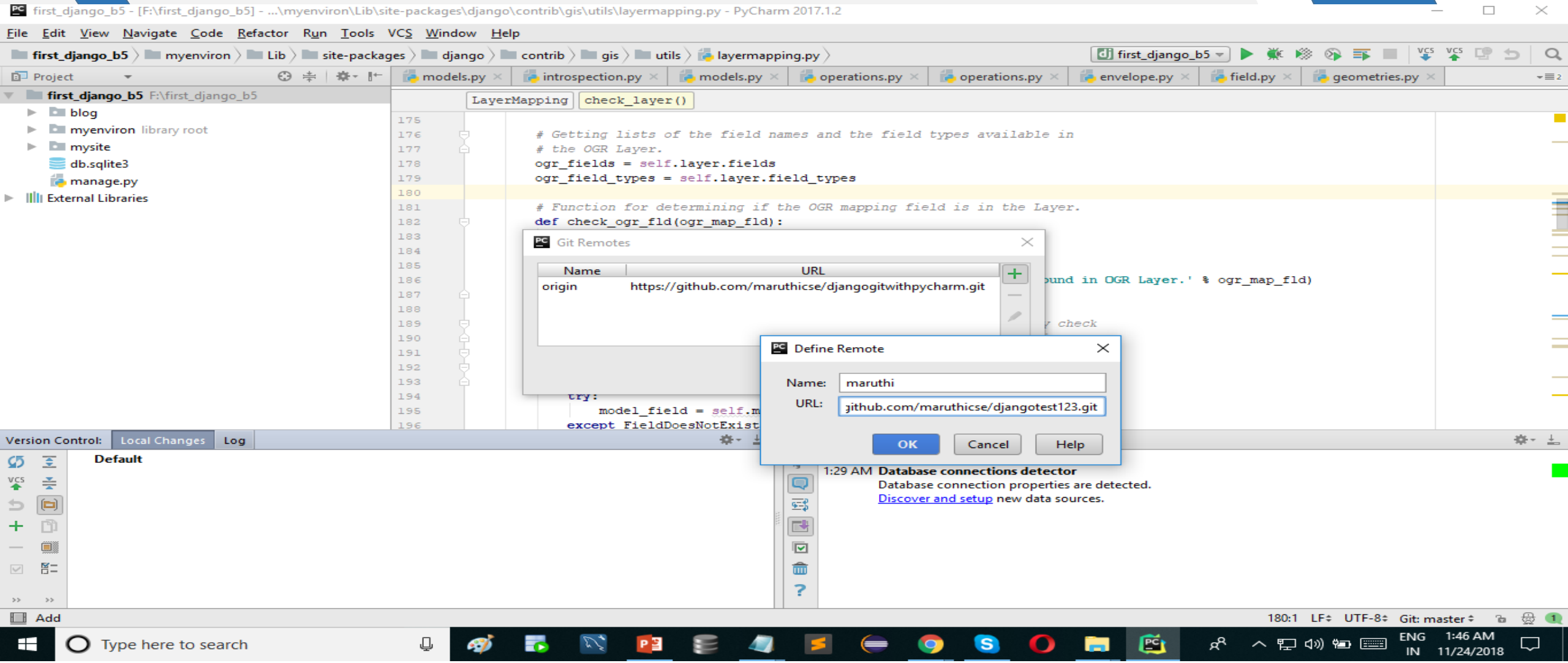
Creating URL in git



Creating URL in git

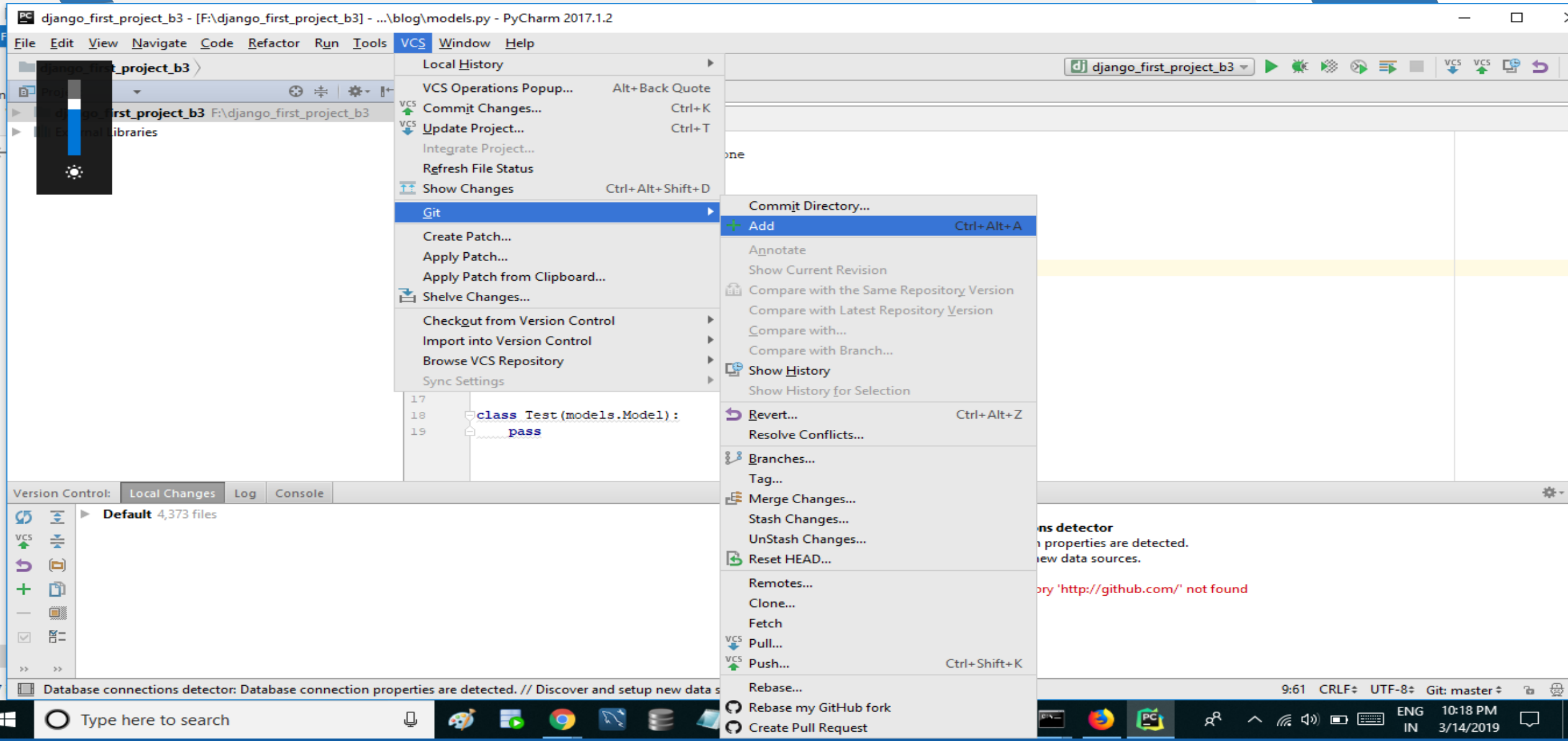


Adding remotes in pycharm



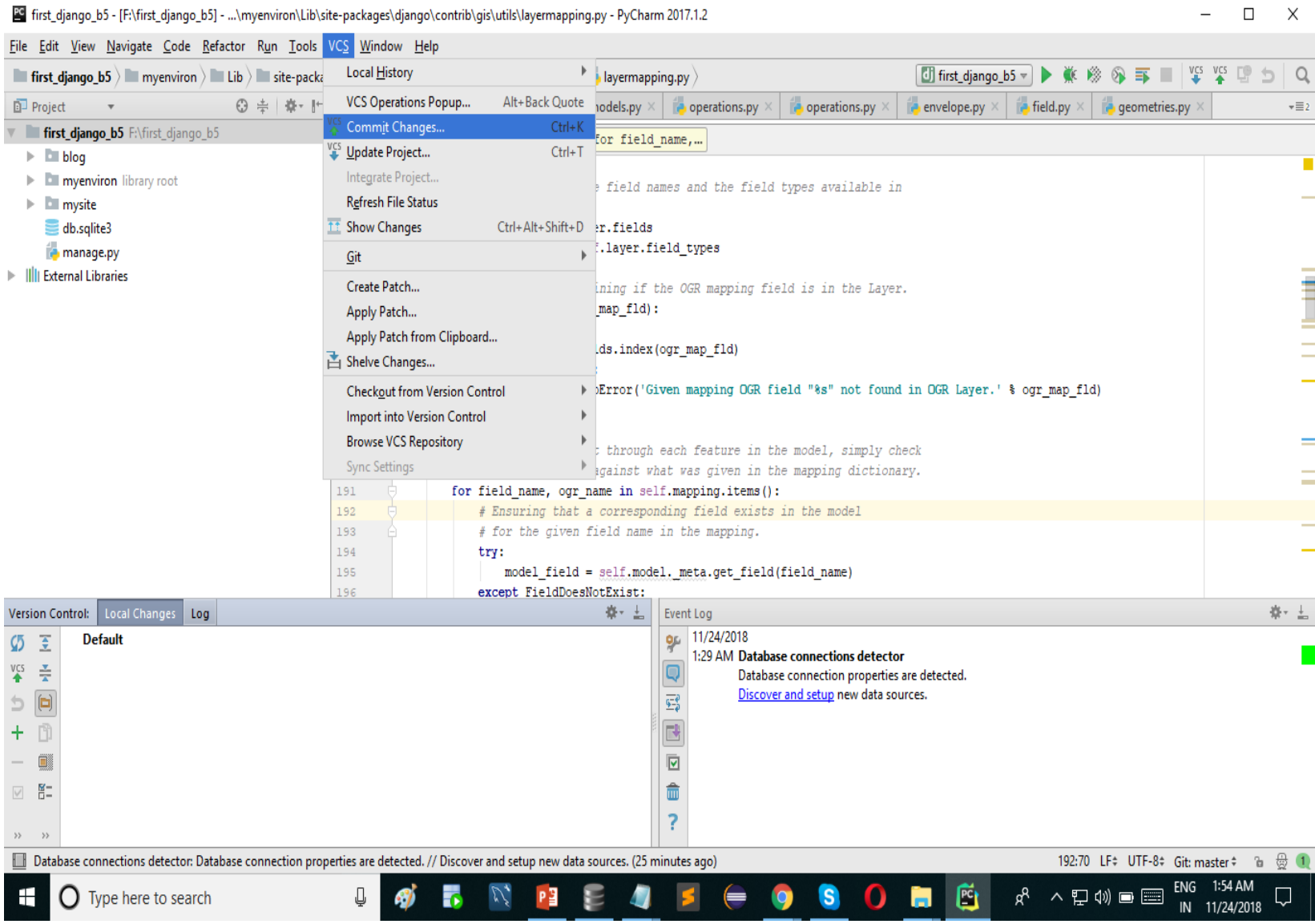
https://www.jetbrains.com/help/pycharm/set-up-a-git-repository.html#put-existing-project-under-Git

Adding files in pycharm

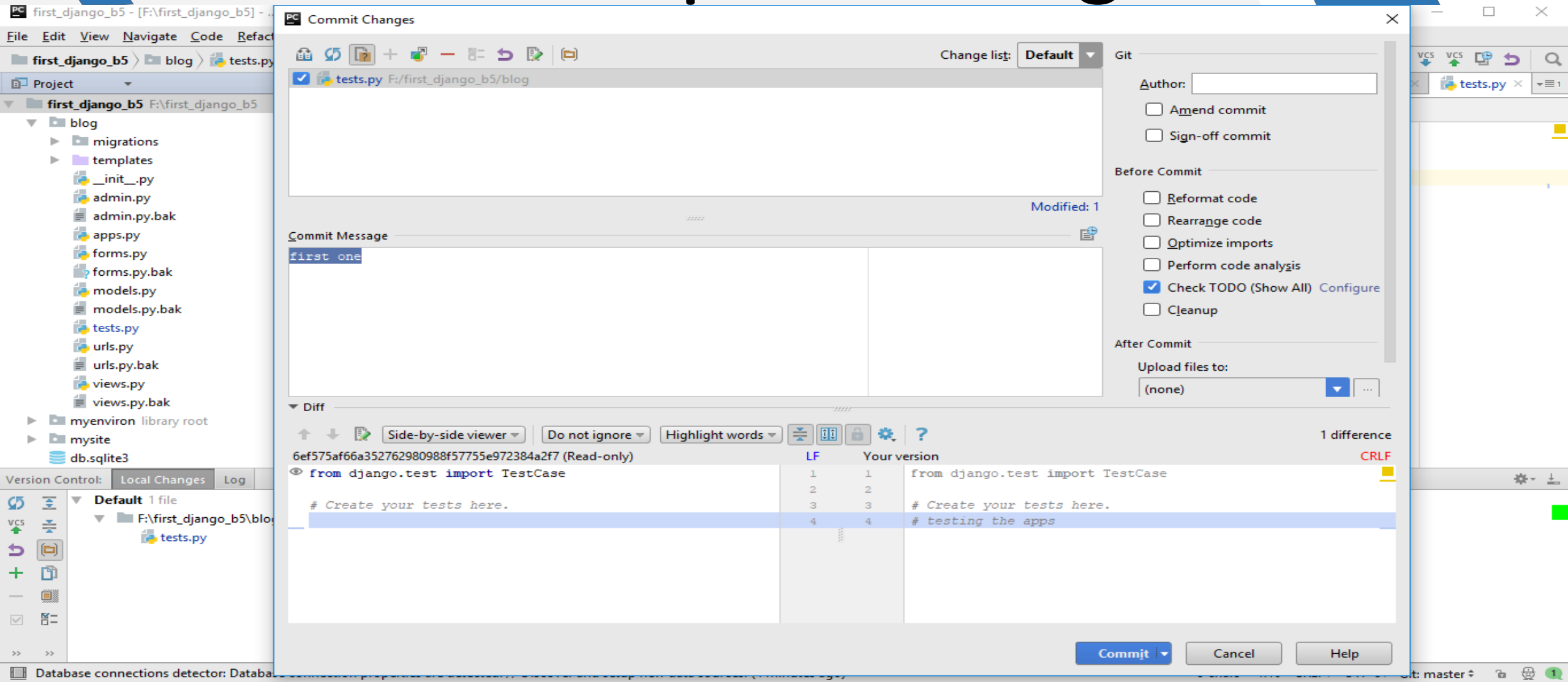


Commit and push changes

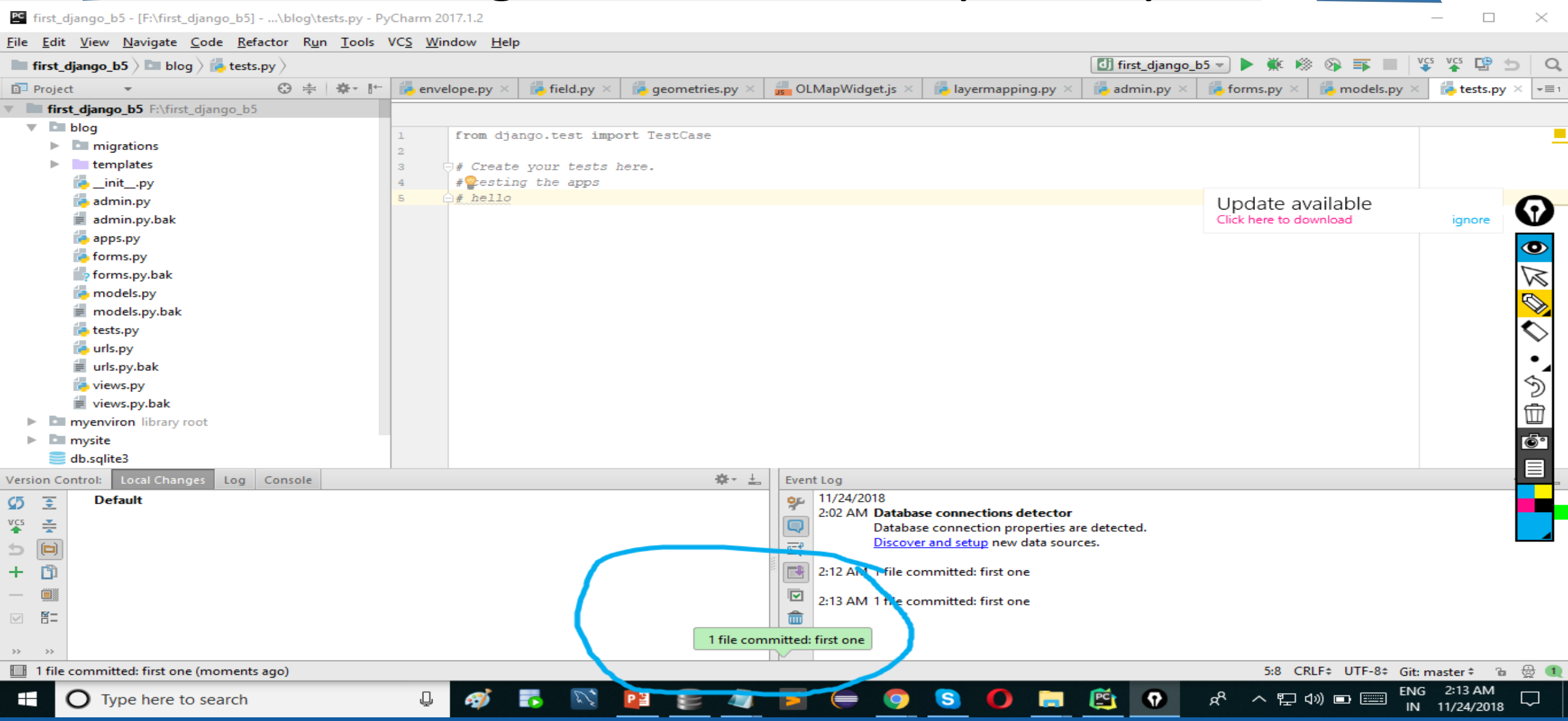
After you've added new files to the Git repository, or modified files that are already under Git version control and you are happy with their current state, you can share the results of your work. This involves committing them locally to record the snapshot of your repository to the project history, and then pushing them to the remote repository so that they become available to others.



Commit and push changes



Push changes to a remote repository



Commit changes locally

- To invoke the Commit Changes dialog, select the files (or an entire changelist) in the Local Changes view and click icons actions commit svg on the toolbar or choose Commit Changes on the context menu of the selection; or press Ctrl+K. The Commit Changes dialog lists all files that have been modified since the last commit, and all newly added unversioned files.
- Enter a commit message and select the Before Commit actions you want IntelliJ IDEA to perform before committing the selected files to the local repository. You can click icons general messageHistory to choose from the list of recent commit messages.
- Select the following options in the Git section if necessary:
  - Author: if you are committing changes made by another person, you can specify the author of these changes.
  - Amend commit: select this option if you want to add the local changes to the latest commit (see Combine staged changes with the previous commit (amend commit) for details).
  - Sign-off commit: Select this option if you want to sign off your commit, i.e. to certify that the changes you are about to check in have been made by you, or that you take the responsibility for the code in question. When this option is enabled, the following line is automatically added at the end of the commit message: Signed off by: <username>

Commit changes locally

- Click the Commit button or hover the mouse over this button to display one of the following available commit options:
  - Commit and Push: push the changes to the remote repository immediately after the commit.
  - You can also press Ctrl+Alt+K to invoke the Commit and Push action from the Commit dialog.
  - Create Patch: generate a patch based on the changes you are about to commit. In the Create Patch dialog that opens, type the name of the patch file and specify whether you need a reverse patch.
  - Remote Run: run your personal build. This option is only available when you are logged in to TeamCity. Refer to TeamCity plugin documentation for details.

Push changes to a remote repository

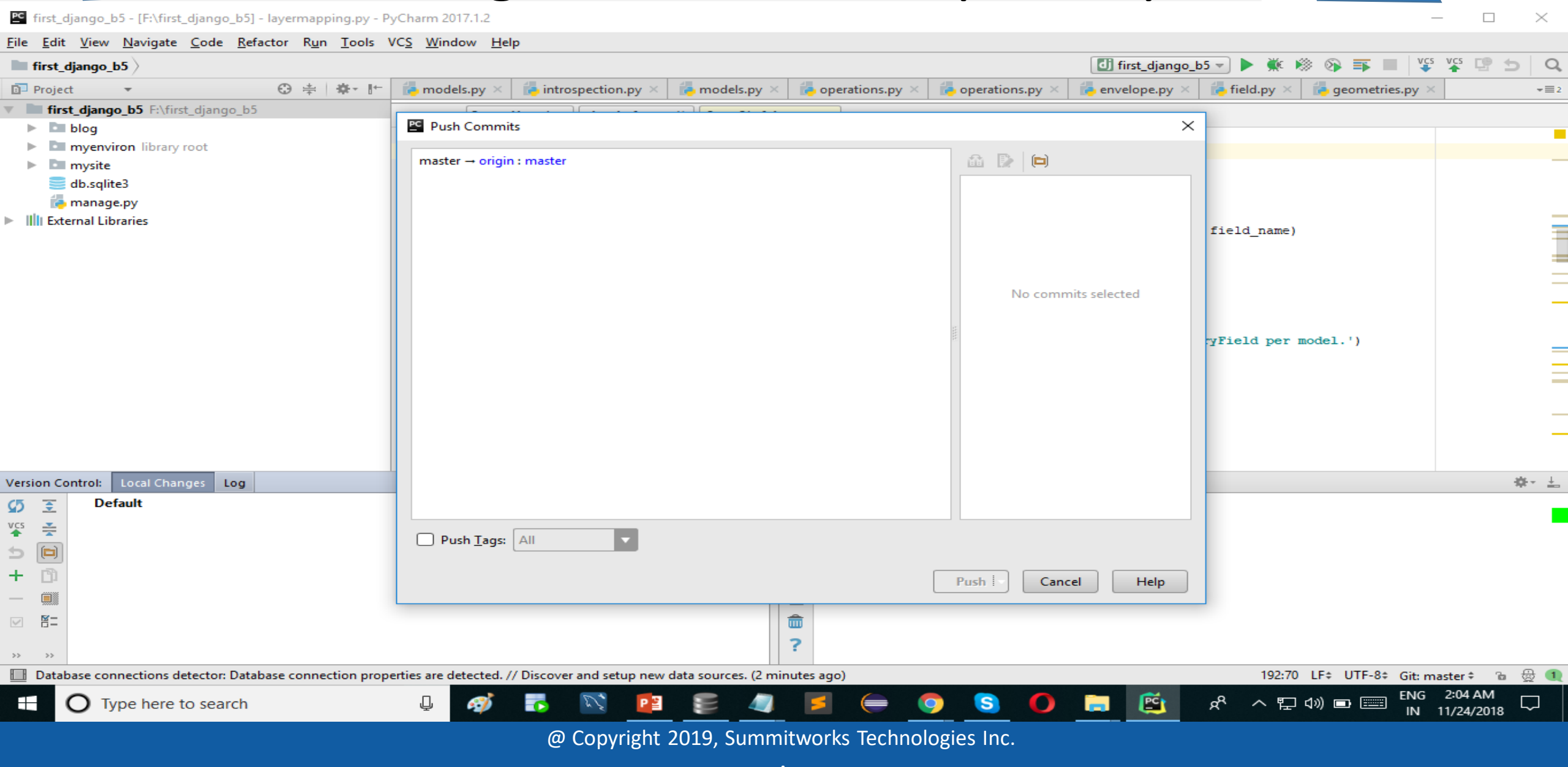
Make sure you have git login credentials.

- Press Ctrl+Shift+K or choose VCS | Git | Push from the main menu. The Push Commits dialog opens showing all Git repositories (for multi-repository projects) and listing all commits made in the current branch in each repository since the last push. If you have a project that uses multiple repositories that are not controlled synchronously, only the current repository is selected by default (for details on how to enable synchronous repositories control, refer to Version Control Settings: Git).

- If there are no remotes in the repository, the Define remote link appears. Click this link and specify the remote name and URL in the dialog that opens. It will be saved and you can edit it later via VCS | Git | Remotes (for details, see Add a remote repository).

- If you want to modify the target branch where you want to push, you can click the branch name. The label turns into a text field where you can type an existing branch name, or create a new branch. You can also click the Edit all targets link in the bottom-right corner to edit all branch names simultaneously. Note that you cannot change the local branch: the current branch for each selected repository will be pushed

Push changes to a remote repository

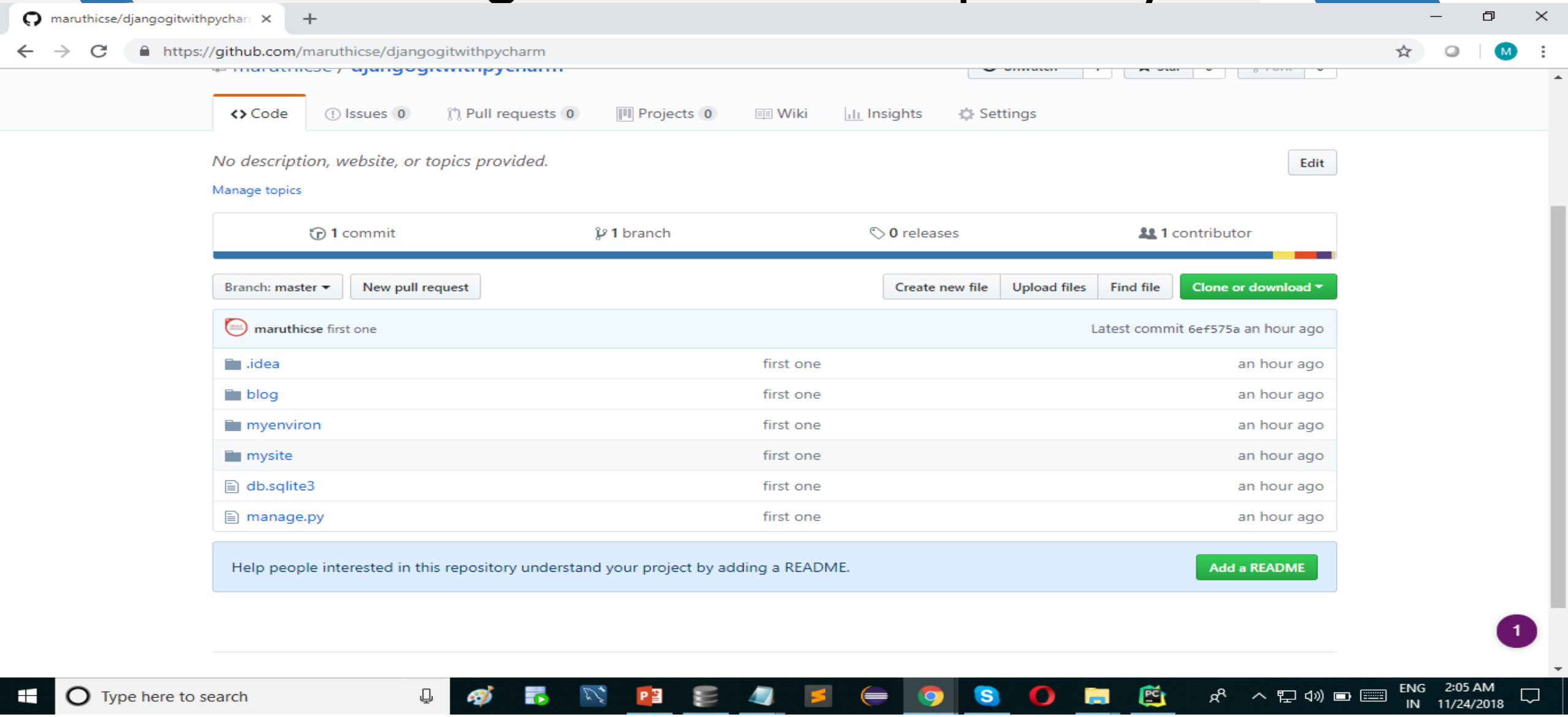


Push changes to a remote repository

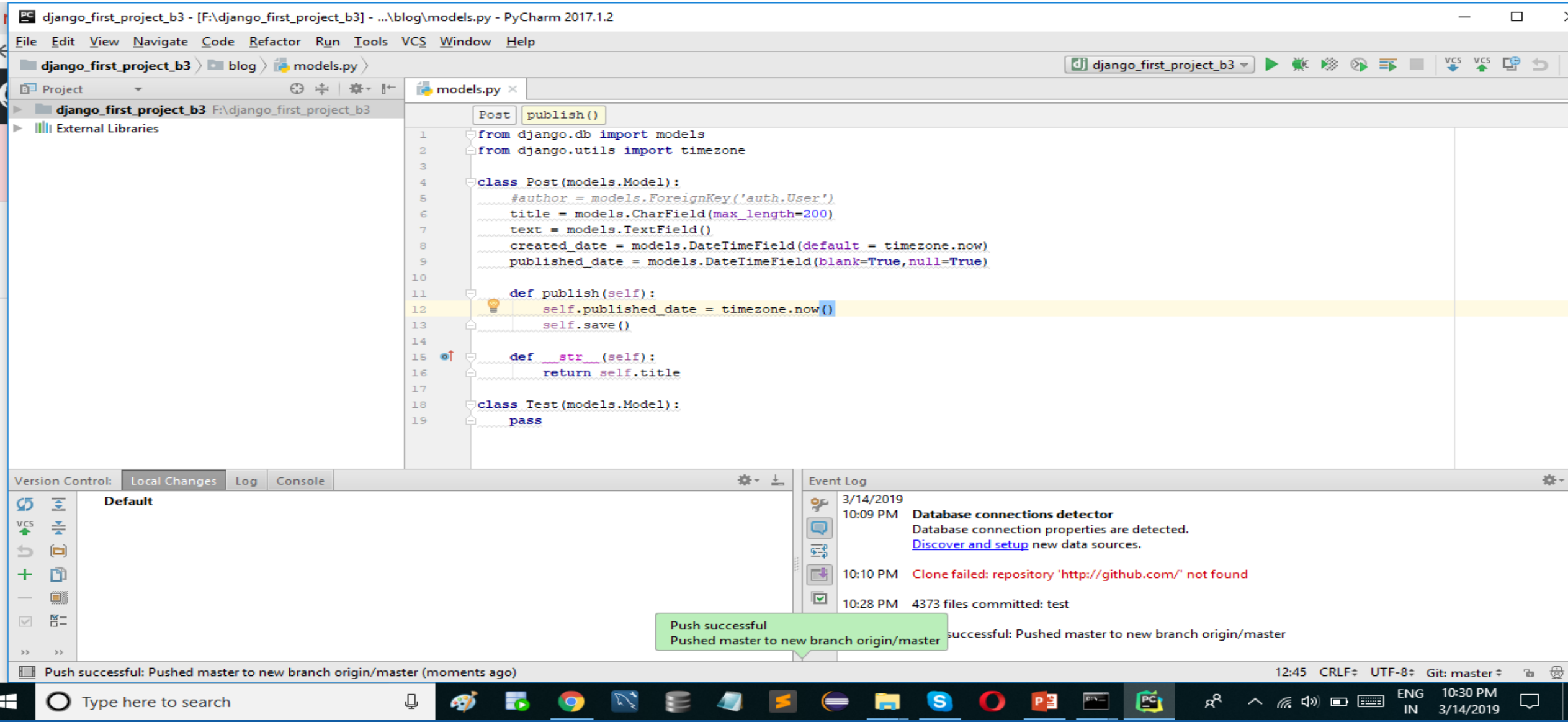
If you want to preview changes before pushing them, select the required commit. The right-hand pane shows the changes included in the selected commit. You can use the toolbar buttons to examine the commit details.

Click the Push button when ready and select which operation you want to perform from the drop-down menu: Push or Force push.
These choice options are only available if the Allow force push option is enabled (see Version Control Settings: Git), otherwise, you can only perform the push operation.

Push changes to a remote repository



Push changes to a remote repository



Push changes to a remote repository

Check list

1. Make sure you have valid git credentials.
2. Create local repo before you send remote repo
3. Create git url for the project before you add remotes for the pycharm project if you are pushing a new project to git
4. Commit changes before you push the code[make sure nothing to commit]
5. Enter commit message while coming changes.
6. Exclude files from version control (ignore) like virtual environment folder.

for more information
https://www.jetbrains.com/help/pycharm/set-up-a-git-repository.html#put-existing-project-under-Git

Pull source from a remote repository

1. $ git clone https://github.com/maruthicse/firstbootcampdjango1
2. $ cd firstbootcampdjango1/
3. You can create virtual environment
4. Install required software's
5. Activate it
6. Run the project

Queries?