Editing the application code

Once your application is created, you will probably want to manage the files (scripts) responsible for execution of the analysis you want to perform. This may be done in several ways, depending on the complexity of your software and on your command of the GIT version control system.

Contents of this guide

With this guide you will learn how to access and edit your application files both from the Application Workbench and from your local computer. Please consult also the Custom application configuration guide for more information about the contents of the descriptor files required for the integration of the application with EPISODES Platform.

Where to look for the application files?

Once you create your own application, it is stored in a repository within the Application Workbench. All your repositories (applications) are listed at the start page of Application Workbench when you log in (if you do not have the Application Workbench account yet, please consult this guide). A sample view is shown on Figure 1 - the right panel contains a list of repositories containing your applications or applications shared with you by other users - one of such repositories is marked in the figure. The first part (here: *tcs-test-user*) of the repository name is the username (yours or the user who shared the repository with you), the second part (here: *TestApp*) is the application name. The central panel of the page is a list of activities related to these applications - e.g. the latest modifications.

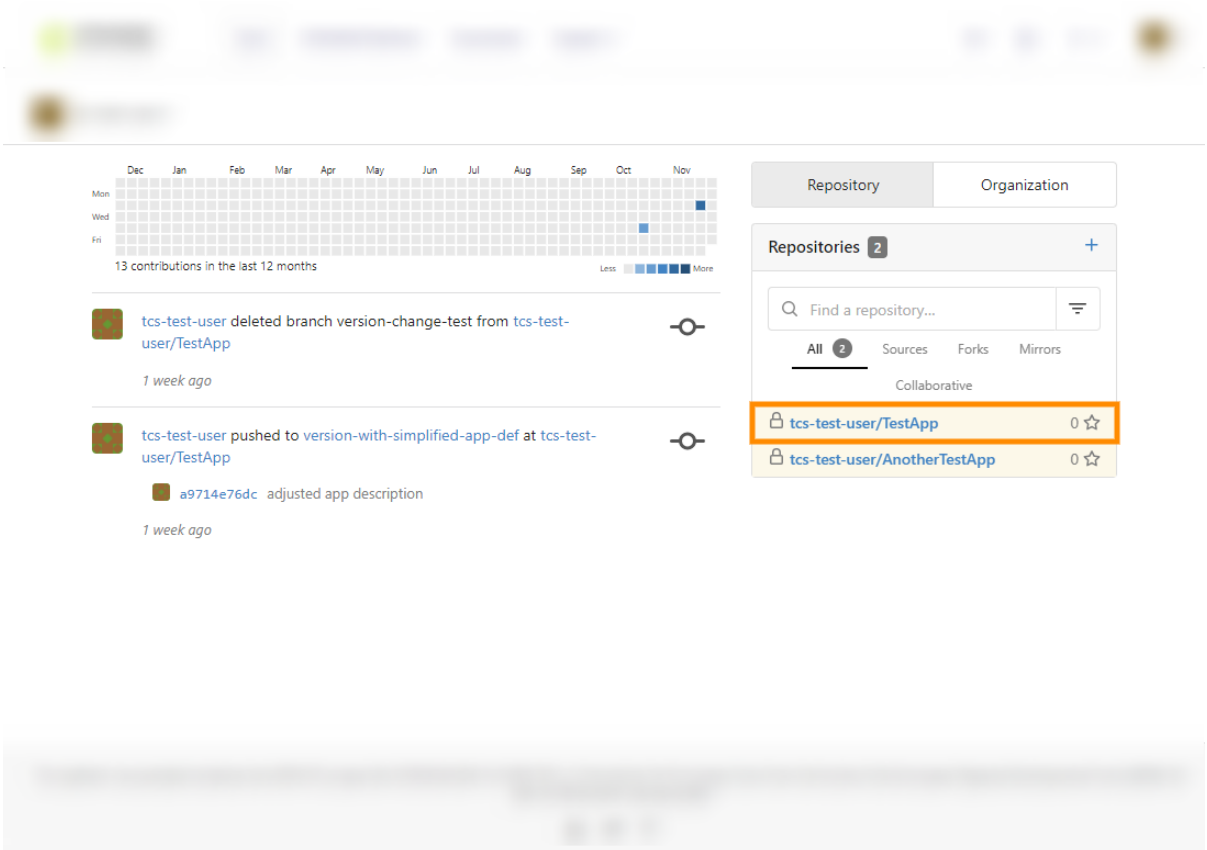

Figure 1. View of the Application Workbench start page, with repository containing *TestApp* application marked

If you are in the EPISODES Platform, you can access directly the code of one of your applications from the *My Apps* page, by choosing the *Code repository* link (see Figure 2). This is equivalent to following the repository link from Figure 1.

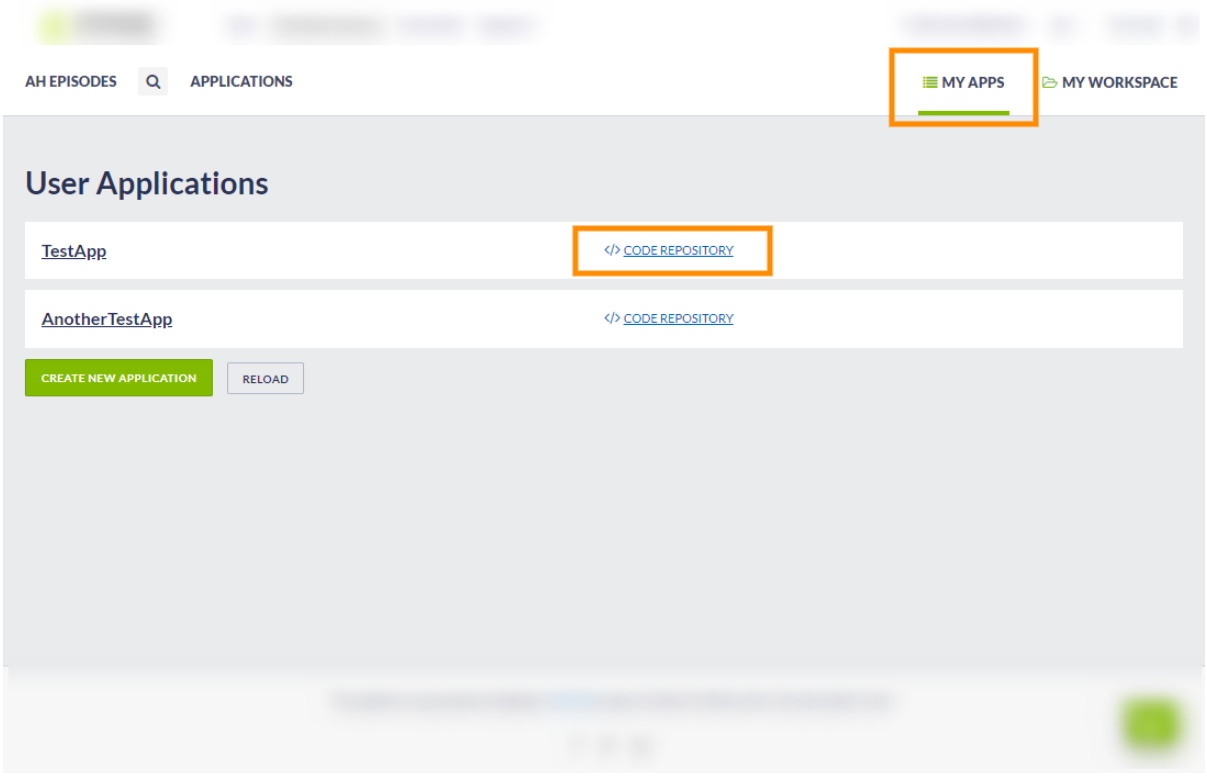

Figure 2. View of the *My Apps* page within the EPISODES Platform.

Editing files through Application Workbench Web interface

The Application Workbench provides you with a Web interface allowing for simple file edition and controlling the modification history. Though the interface provides content assistance and other features that facilitate programming, it is still a simple text editor, therefore, if your application requires advanced programming constructions or uses many files, it is recommended to edit its files on your computer and publish the changes to the Application Workbench from there (this mode of edition is described in Editing files from your own computer section). In this way, you will be able to use your favorite IDE (Integrated Development Environment), e.g., MATLAB Workbench for editing files and to publish changes to multiple files at once. Obviously, the same effect can be reached using the Web interface, however, it might be less comfortable in the aforementioned cases.

Contents of the application's repository

Once you enter the repository containing your application, a page containing its main directory will be displayed (Figure 3). If you used the EPISODE Platform's wizard for creating the application, the repository will be already populated with files automatically generated by the Platform - the files are represented by rows in the central part of the page, with the file name on the left (marked with **(1)** in Figure 3) and the comment explaining the last modification and date of this modification displayed on the right (marked with **(2)** in Figure 3). Note, that if you created the application repository manually, the repository will be empty or it will contain only a `README.md` or license file (depending on your chosen settings), but it will not contain the configuration and execution files required by the EPISODES Platform to use the application - you will have to add them manually. Adding a file is possible using the *New file* or *Upload file* button (marked with **(3)** in Figure 3). Choosing the *New file* button will result in the system showing an edit page (as in case of editing a file - see Figure 5 and further paragraphs). Note, that it is not possible to create a directory in this way, a directory can be created only implicitly by editing a file name (see the next subsection) or by creating a nested directory structure on your computer and publishing it to the repository (see Editing files from your own computer section).

To access the application's repository settings (e.g. add collaborators, that will be able to see/edit your application), use the *Settings* button (mark ed with **(4)** in Figure 3).

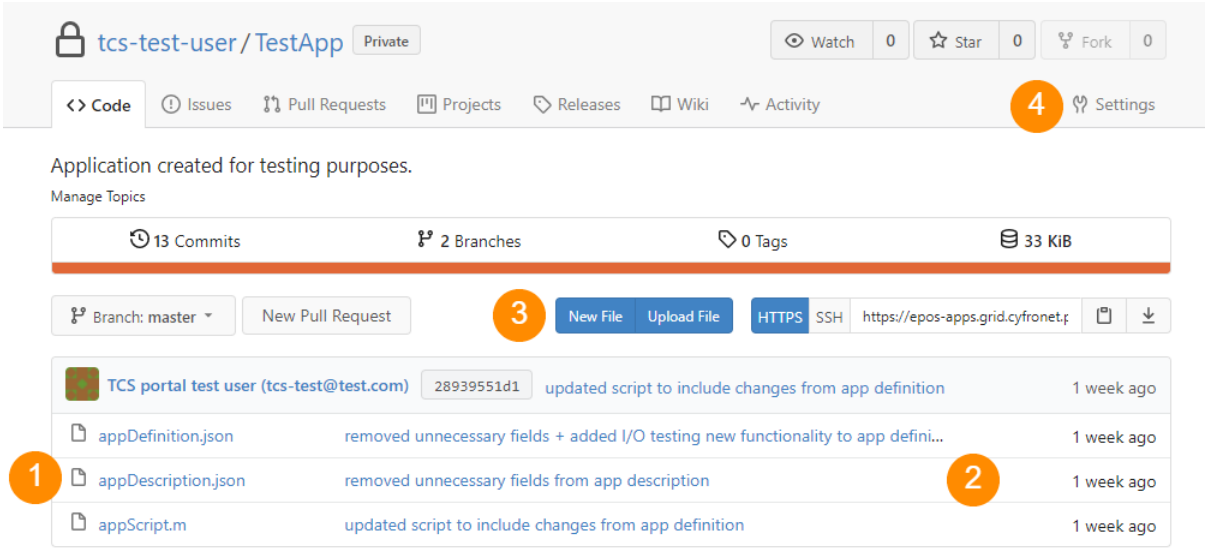

Figure 3. Main view of the repository containing the *TestApp* files, with most important elements marked

When entering any of the files (by clicking the file name - **(1)** in Figure 3), a view similar to Figure 4 is displayed, showing the file contents and controls. In case of entering a directory, the contents of the directory (files) are displayed similarly to the file listing shown in Figure 3. The pencil icon (marked with **(1)** in Figure 4) enables editing the file (see Figure 5), the trash icon (marked with **(2)** in Figure 4) allows for deleting the file, and the button *History* (marked with **(3)** in Figure 4) shows the history of modifications of the file.

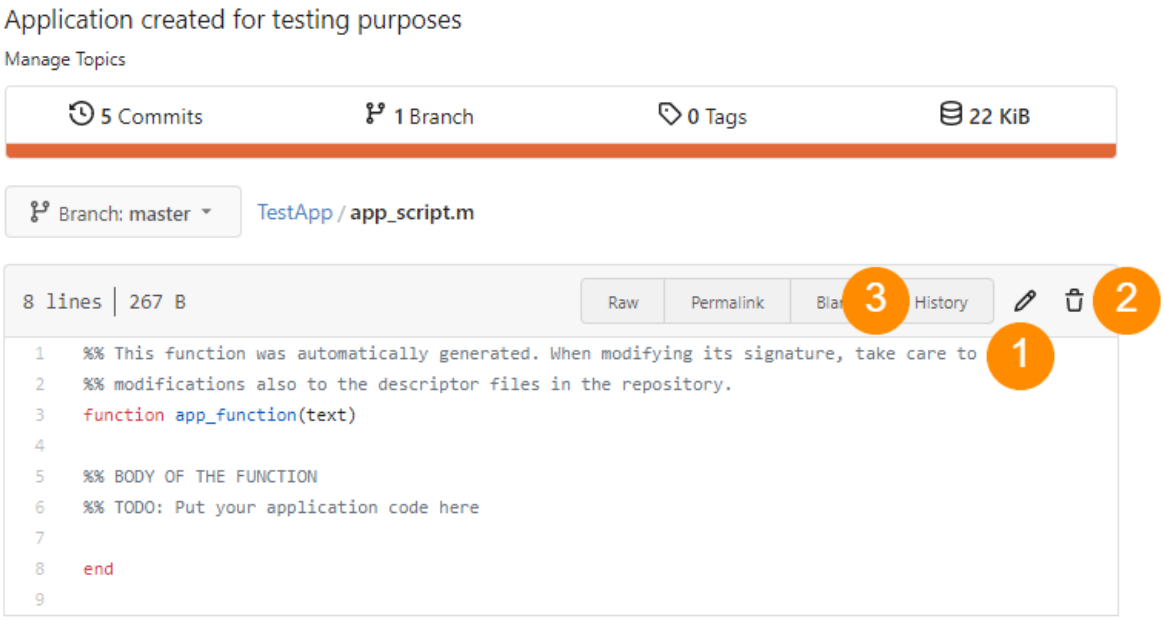

Figure 4. View of chosen file contents, with most important elements marked. Here: the file `app_script.m` from the *TestApp* application

It is also possible to view a raw version (without any additional options and line numbering), by using the *Raw* button (Figure 4). The file can then be downloaded by using the Web browser's option *Save as...* (CTRL + S). Creating a permanent link to the file view can be achieved with the button *Permalink*. Note, that this will create a link to the exact version of the file you are viewing at the moment.

Editing files and saving changes

Once in the editing mode, the text editor allows for any modifications to the chosen file. Continuing with the example displayed from the previous section (Figure 4), we changed the contents of the `app_script.m` file adding the body of the function *app_function*. This is done in the text editor - marked with **(1)** in Figure 5. The changes can be checked within the *Preview Changes* tab (marked with **(2)** in Figure 5) - with an effect as shown in Figure 6 - the changed lines are marked with colors. To change the name of the edited file, use the field marked with **(3)** in Figure 5 (the current file name is already inserted there). In this way, you can also create a directory to put the file in - by inserting a directory name, followed by the `/` (slash) path separator before the file name. E.g., to put the `app_script.m` in a directory `dir`, we would need to change the content of this field to `dir/app_script.m`. Note, that it is not possible to create an empty directory in this way.

Saving the changes both to the file name and its contents is realized with the *Commit Changes* button (marked with **(5)** in Figure 5). It is also advisable to add a comment when saving the changes (field marked with **(4)** in Figure 5), describing the change, and, possibly, its purpose. If no comment is provided, a default "*Update '<filename>*" (here it would be "*Update 'app_script.m*") is inserted by the system. The comment related to the latest version of a file is later visible in the directory listing view (see marking **(2)** in Figure 3). It will be also visible in the file history, therefore, if you comment each change, it will be easier four you to later analyze the history.

Note for those already familiar with the GIT versioning system: the *Commit Changes* in this view in fact performs both `git commit` and `git push` actions.

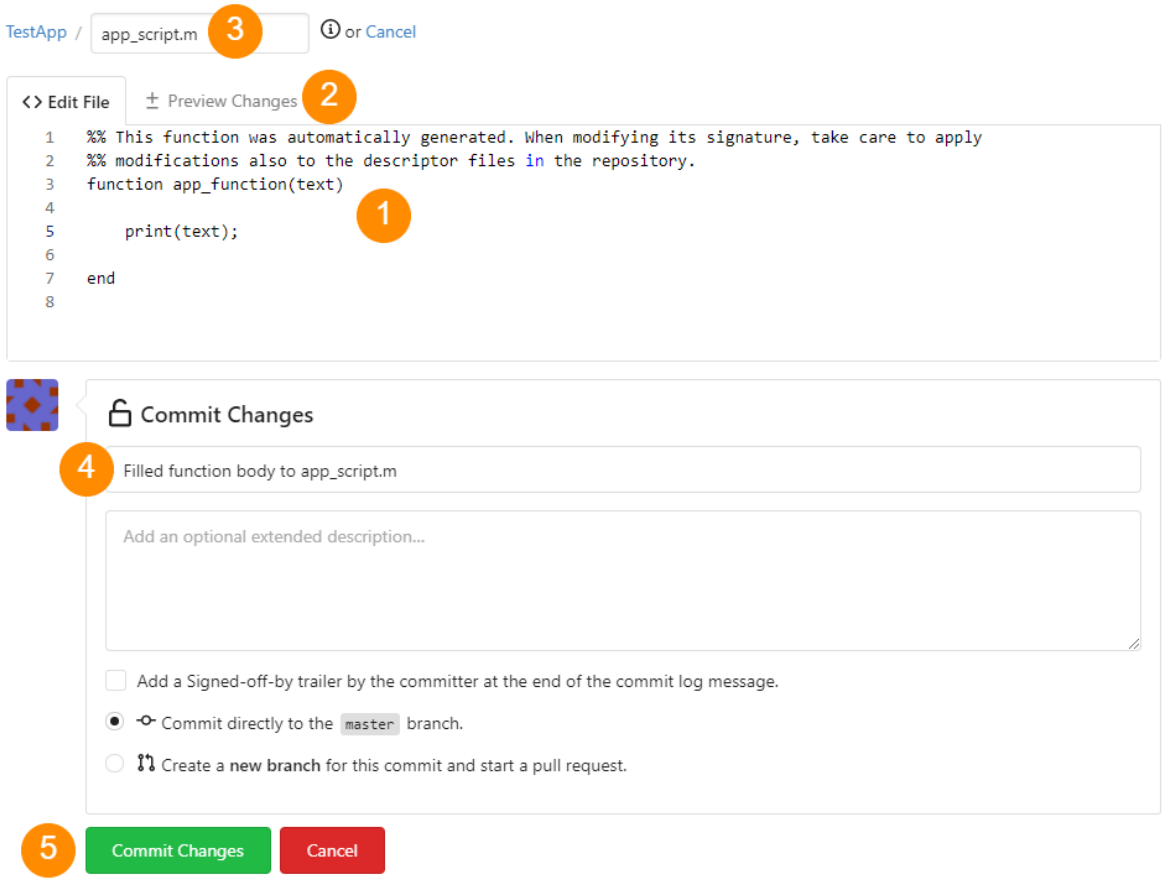

Figure 5. Edition view of the *app_script.m* file, with text editor and other most important elements marked

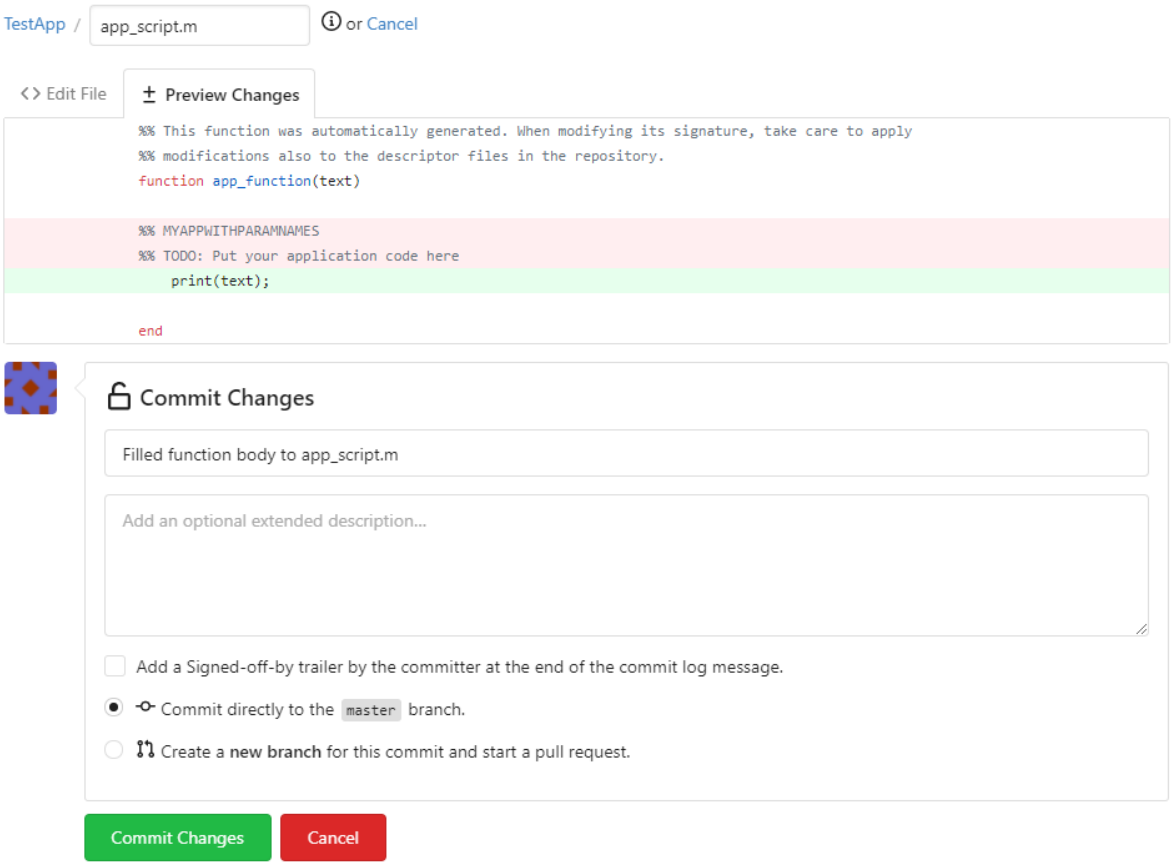

Figure 6. Edition view of the *app_script.m* file, with change preview

Note, that, apart from the text editor, the other controls of the Preview Changes tab are the same as in the Edit File tab, therefore, you can perform all the operations except for the file contents editing on any of the two tabs.

Editing files from your own computer

Prerequisites

Publishing changes applied on your local filesystem to the application's repository in the Application Workbench has to be done using the GIT versioning system. For this purpose, it has to be installed on your local computer. The GIT command-line client can be downloaded from http://git-scm.com/downloads, but there are also several GUI tools facilitating work with GIT - some of them are listed at http://git-scm.com/downloads/guis. Many of the programming environments (IDEs) also offer support for work with GIT, either as a built-in feature or as a separate plug-in. Here, we will provide instructions using GIT commands, for GUI-based tools, a tool-specific controls should be used, however, they should be named similarly as the commands used here. We advise also to read the GIT Getting Started guide if you are new to this versioning system.

Unless you already imported your application's repository from your computer, you will first have to download it to a location of your choice. For that, you will need to clone the repository to your computer by invoking `git clone <repository_cloning_url>` in a directory where you will want to store the code of the application. The repository cloning URL will have a form of: `https://epos-apps.grid.cyfronet.pl/<username>/<app_name>.git` - in the example from this page it would be `https://epos-apps.grid.cyfronet.pl/tcs-test-user/TestApp.git`. It can found in the repository view within the Application Workbench, as marked on Figure 7 (next to the URL there is an icon that allows to copy it to clipboard). More information about cloning repositories can be found in this GIT guide.

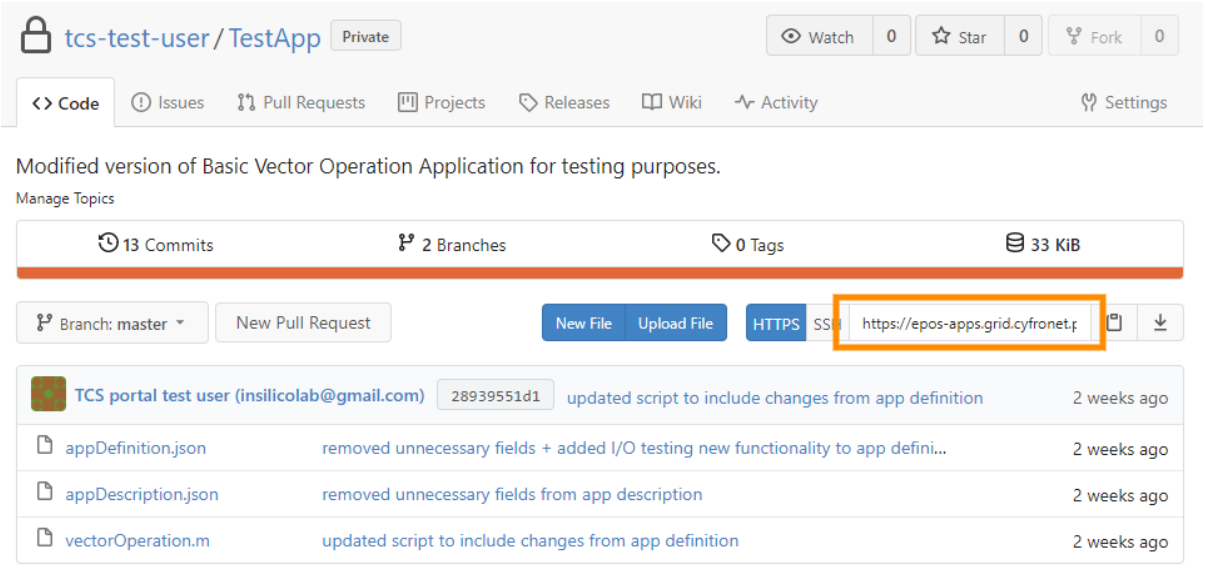

Figure 7. View of the repository in Application Workbench, with the repository cloning URL marked

The cloning operation initializes all the required GIT structures in the chosen directory (you will notice a `.git` direcory created inside, which will store all the required metadata) and download all the files you already had in the repository.

At this stage, you can modify the files and create new at your will inside the local application folder. The modifications in GIT are done in three phases: 1) staging the files, which selects the chose modifications for later publication, 2) committing them - a commit groups all the selected modifications as one package (this operation is still local), 3) pushing the files to the remote location (this is a final publication of the changes to the repository in Application Workbench). Therefore, for each of t he files that was changed or added, use `git add <filename>` (regular expressions for adding more files at a time are also supported). When the changes are complete, use `git commit -m "<modification comment>"` to create a package of changes that will be later published. The modification comment should include a general description, and, advisably, purpose of the changes you are performing. The comment related to the latest version of a file is later visible in the directory listing view (see marking **(2)** in Figure 3) in the Application Workbench and in the file history. If no comment is added (no `-m` parameter), a text editor will be opened to type it. Note, that at this stage, your changes are not yet published. To publish them to the remote repository (in Application Workbench), use `git push`. Note, that you do not need to push after each commit, you can group different changes in different commits and then push them all at once. However, you will not see any modifications in your application view in Application Workbench nor in EPISODES Platform until you do the push. After publishing the changes, it is advisable to check the effect in the Application Workbench Web interface (see th e previous section). You can also complement the local modifications with small edits performed through that interface.

For more information on working with files in GIT, see this GIT guide. You can also get some help on the subsequent actions by typing `git status` in the command line at any moment.

Related Documents

- Application Definition file
- Creating Application Workbench account
- Running your custom application
- Custom application configuration
- Creating Custom Applications