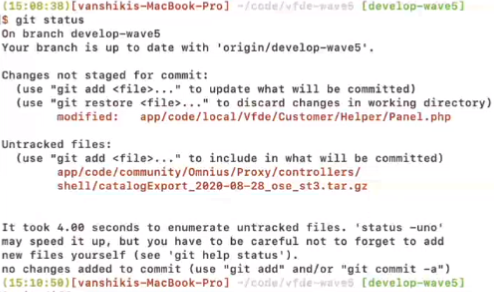
**PULL REQUEST STRATEGY**

1. Switching the branch where we need to fix bugs or create functions

Git Checkout - To start working in a different branch, use *git checkout* to switch branches.

Use: - Here switching to develop-wave5

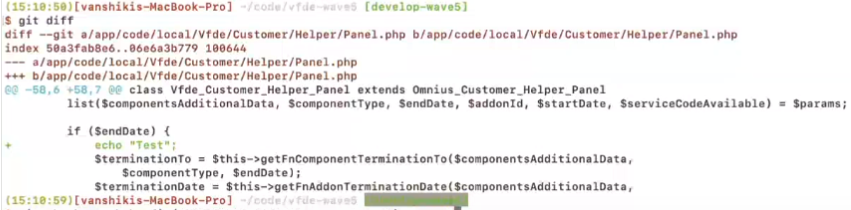
1. Checking the status of the Repository

Git Status- This command returns the current state of the repository. Git status will return the current working branch. If a file is in the staging area, but not committed, it shows with git status. Or, if there are no changes it’ll return nothing to commit, working directory clean.

Use: - Here we can see the file Panel.php is highlighted in red as it is not committed.

1. Checking the difference

Git Diff- It is used in git to track the difference between the changes made on a file. Diff command takes two inputs and reflects the differences between them. It is not necessary that these inputs are files only.

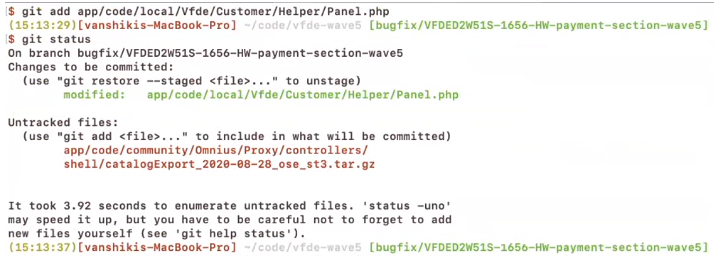


1. Create bugfix branch - bugfix/ticket\_number-Section-Wave-5



Use: - Created branch for given ticket number.

1. Adding the file to the branch and checking the status for file that we have added.

Git Add- Adds files in the to the staging area for Git. Before a file is available to commit to a repository, the file needs to be added to the Git index (staging area). There are a few different ways to use git add, by adding entire directories, specific files, or all unstaged files.

Use: - Added the Panel.php file

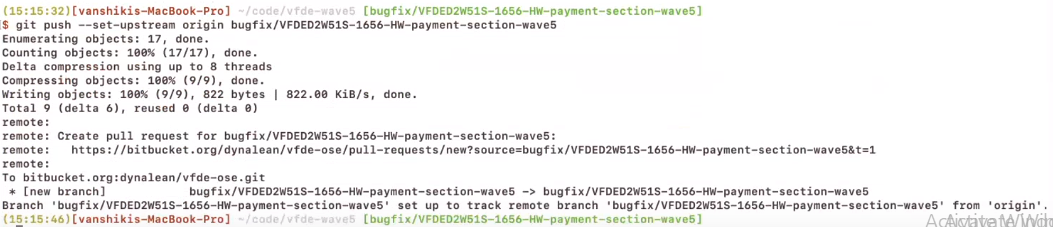
1. Committing the file to the branch

Git commit- Record the changes made to the files to a local repository. For easy reference, each commit has a unique ID.

It’s best practice to include a message with each commit explaining the changes made in a commit. Adding a commit message helps to find a particular change or understanding the changes.

Use: - Committed with the message using tag -m

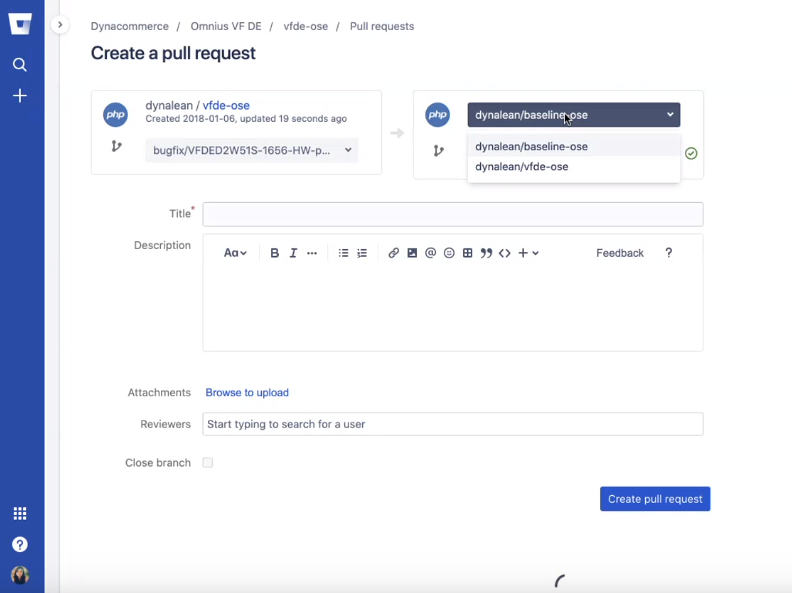
1. Pushing the file to the branch

Git Push- Sends local commits to the remote repository. Git push requires two parameters: the remote repository and the branch that the push is for.

Use: - Here we got the remote link for PR

1. Link that we got from previous step

Fill particular wave branch in the given space and must read all the instructions to specifying labels and attachments in the given space.



Follow PSR2 standard coding norms:

* Code MUST follow a “coding style guide” PSR [[PSR-1](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-1-basic-coding-standard.md)].
* Code MUST use 4 spaces for indenting, not tabs.
* There MUST NOT be a hard limit on line length; the soft limit MUST be 120 characters; lines SHOULD be 80 characters or less.
* There MUST be one blank line after the namespace declaration, and there MUST be one blank line after the block of use declarations.
* Opening braces for classes MUST go on the next line, and closing braces MUST go on the next line after the body.
* Opening braces for methods MUST go on the next line, and closing braces MUST go on the next line after the body.
* Visibility MUST be declared on all properties and methods; abstract and final MUST be declared before the visibility; static MUST be declared after the visibility.
* Control structure keywords MUST have one space after them; method and function calls MUST NOT.
* Opening braces for control structures MUST go on the same line, and closing braces MUST go on the next line after the body.
* Opening parentheses for control structures MUST NOT have a space after them, and closing parentheses for control structures MUST NOT have a space before.

1. Checking for Unit Test Cases and Jenkins Pipelines

All dependencies will be installed and coding standards will be checked

