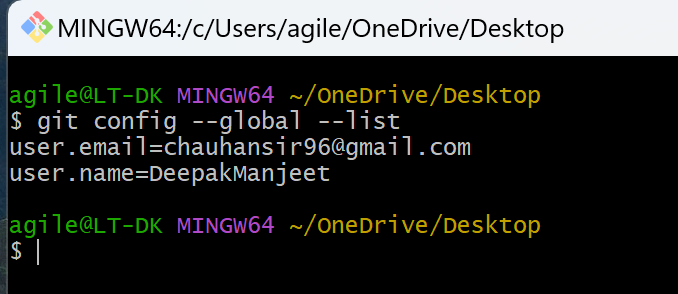
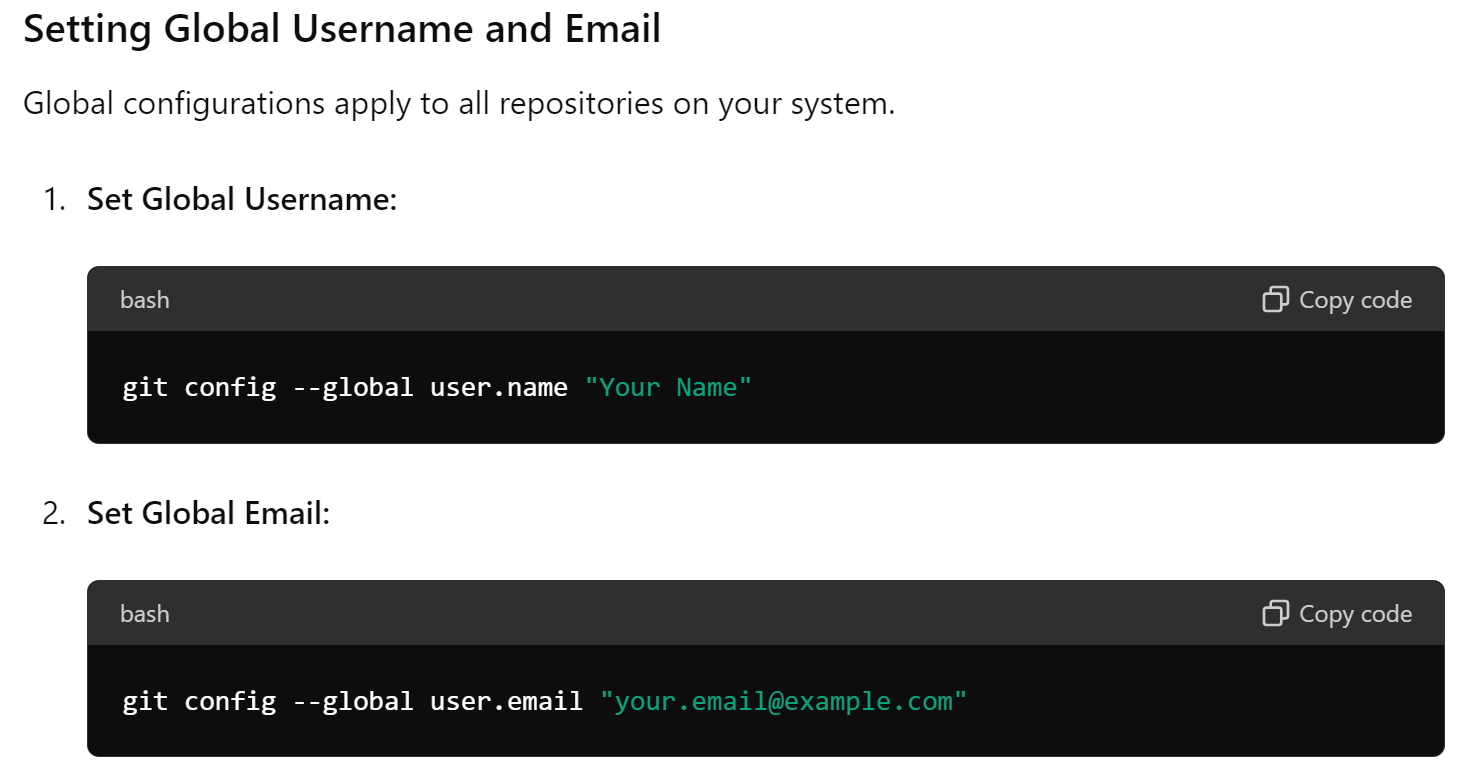
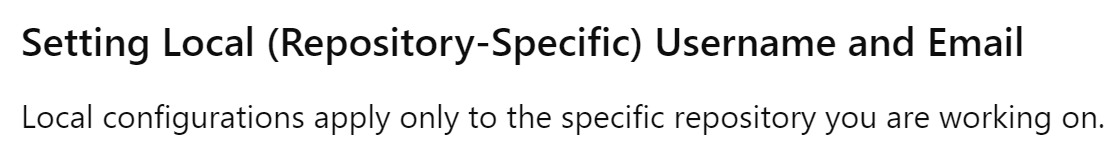


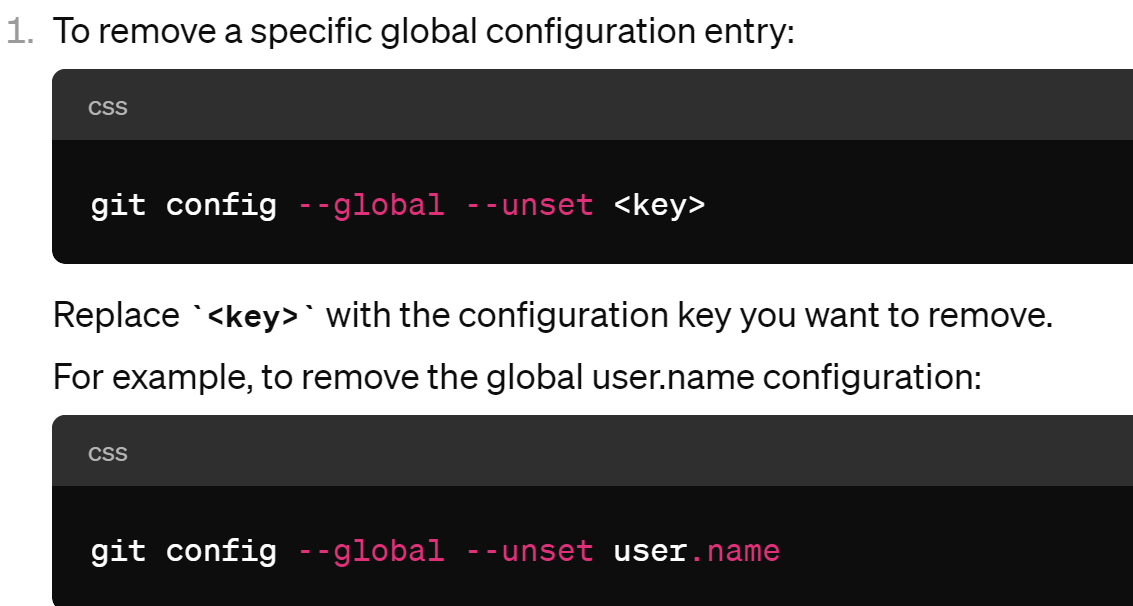
Ex











**Git steps to use in FIS environment:**

1. Apply proxy

git config --global https.proxy <https://e#:pswd@proxy.fnfis.com:8080>

git config --global http.proxy http://e#:pswd@proxy.fnfis.com:8080

* In fisdev.local environment like TA8 above lines are OK, but for client.local environment like WEBSTER, CFC:

**Proxy.fnfis.com** should be replaced by **proxy.prod.local**

|  |  |
| --- | --- |
| **In fisdev.local environment like TA8** | **in client.local environment like WEBSTER, CFC** |
| git config --global https.proxy <https://e#:pswd@proxy.fnfis.com:8080>  git config --global http.proxy <http://e#:pswd@proxy.fnfis.com:8080>  in above lines:  e# = e5721153  pswd = P@ssw0rd@9992284742 (means pswd of system) | git config --global https.proxy <https://e#:pswd@proxy.prod.local:8080>  git config --global http.proxy <http://e#:pswd@proxy.prod.local:8080>  in above lines:  e# = e5721153  pswd = Vista#24Automation (means admin pswd of server) |

Proxy can not be resolved if Password contains special character, so you have to replace those special characters with corresponding % symbol e.g.

1. **Space**: %20
2. **Exclamation mark (!)**: %21
3. **Double quote (")**: %22
4. **Hash or pound sign (#)**: %23
5. **Dollar sign ($)**: %24
6. **Percent sign (%)**: %25
7. **Ampersand (&)**: %26
8. **Single quote (')**: %27
9. **Left parenthesis (()**: %28
10. **Right parenthesis ())**: %29
11. **Asterisk (\*)**: %2A
12. **Plus sign (+)**: %2B
13. **Comma (,)**: %2C
14. **Hyphen (-)**: %2D
15. **Period or full stop (.)**: %2E
16. **Forward slash (/)**: %2F
17. **Colon (:)**: %3A
18. **Semicolon (;)**: %3B
19. **Less than sign (<)**: %3C
20. **Equals sign (=)**: %3D
21. **Greater than sign (>)**: %3E
22. **Question mark (?)**: %3F
23. **Commercial at (@)**: %40
24. **Left square bracket ([)**: %5B
25. **Backslash (\)**: %5C
26. **Right square bracket (])**: %5D
27. **Caret (^)**: %5E
28. **Underscore (\_)**: %5F
29. **Grave accent (`)**: %60
30. **Left curly brace ({)**: %7B
31. **Vertical bar or pipe (|)**: %7C
32. **Right curly brace (})**: %7D
33. **Tilde (~)**: %7E

Thus,

git config --global https.proxy [https://e5721153:P**%40**ssw0rd**%40**9992284742@proxy.fnfis.com:8080](https://e5721153:P%40ssw0rd%409992284742@proxy.fnfis.com:8080)

git config --global http.proxy [http://e5721153:P**%40**ssw0rd**%40**9992284742@proxy.fnfis.com:8080](http://e5721153:P%40ssw0rd%409992284742@proxy.fnfis.com:8080)

1. After setting these proxy, if you clone the project, a window dialog will pop up asking for your bitbucket credentials:

Username = e5721152

Pswd = P@ssw0rd@9992284742

1. You can check already set proxy details:

git config --global --get http.proxy

git config --global --get https.proxy

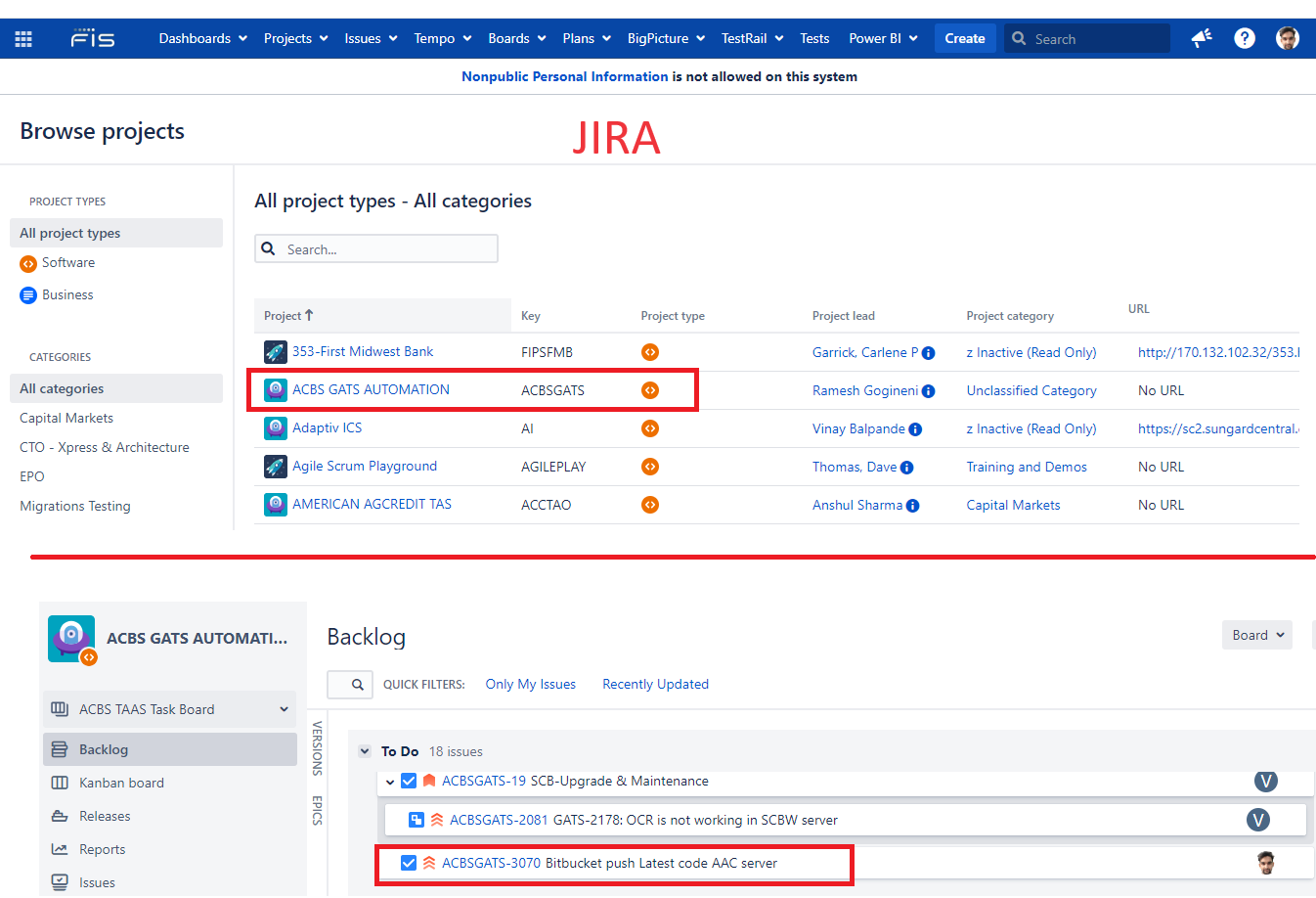
1. You can remove already set proxy settings:

git config --global --unset http.proxy

git config --global --unset https.proxy

**Steps to push code to Bitbucket and merge to master:**

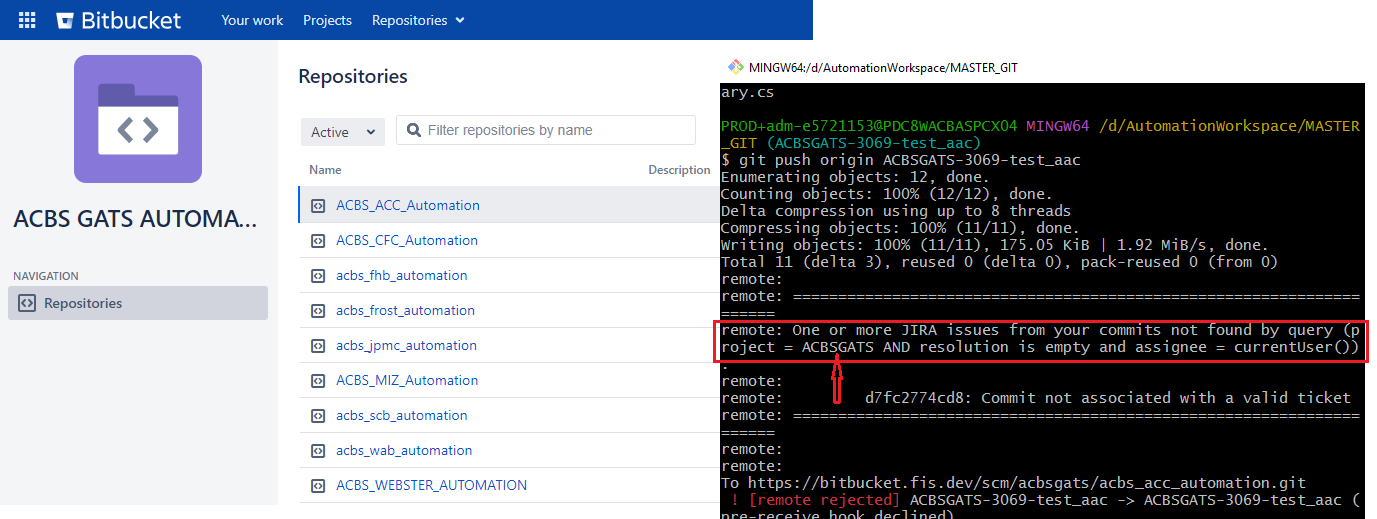
1. Create a Jira ticket under ACBS GATS AUTOMATION project:



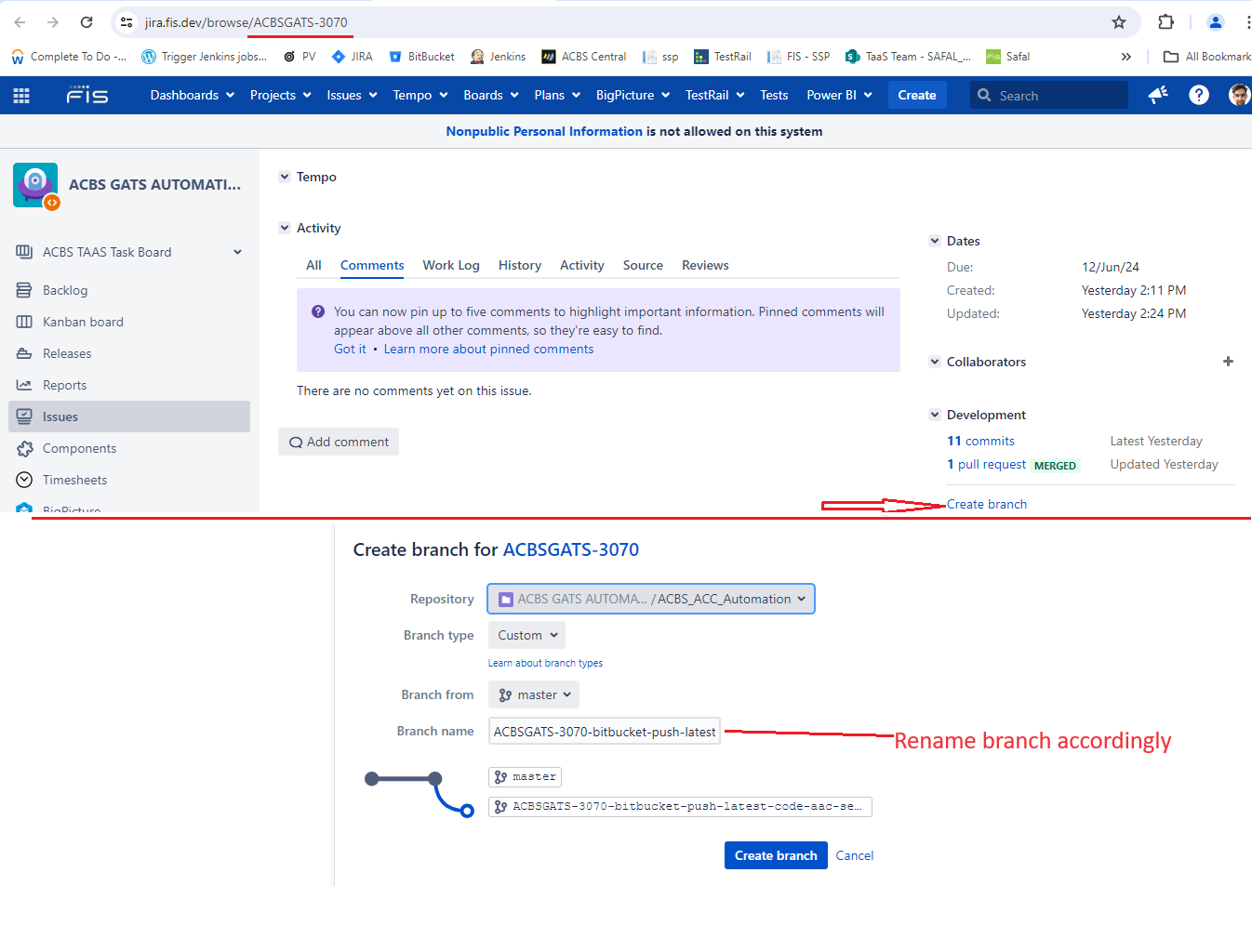
Under this ticket, we will create a branch

We have created this ticket under ACBS GATS AUTOMATION because all our repos are under this project and while pushing code to bitbucket, it executes a query to check for this project name else it will throw error as shown below in right image.

You may change this query by contacting repo admin of bitbucket.

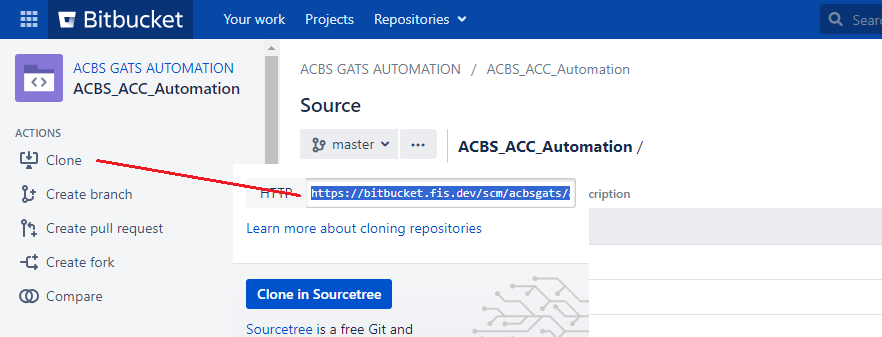


1. Create a branch under this ticket and switch to that branch using git bash on asp server:



1. Now, on the server, in git bash:

* Open git bash in your working directory and clone using master repo link from bitbucket



**git clone above\_link**

* fetch all refs in git on server:

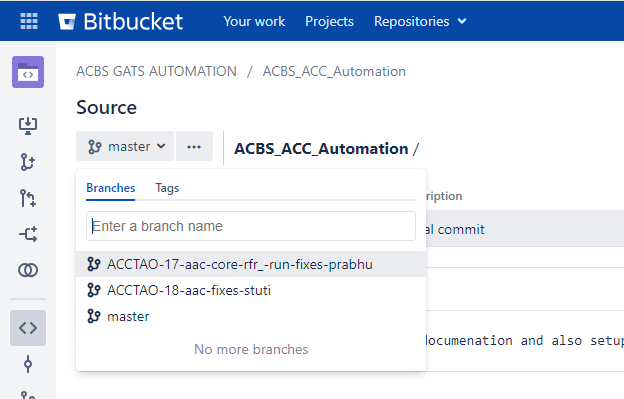
**git fetch --all**

* switch to branch created in jira ticket.

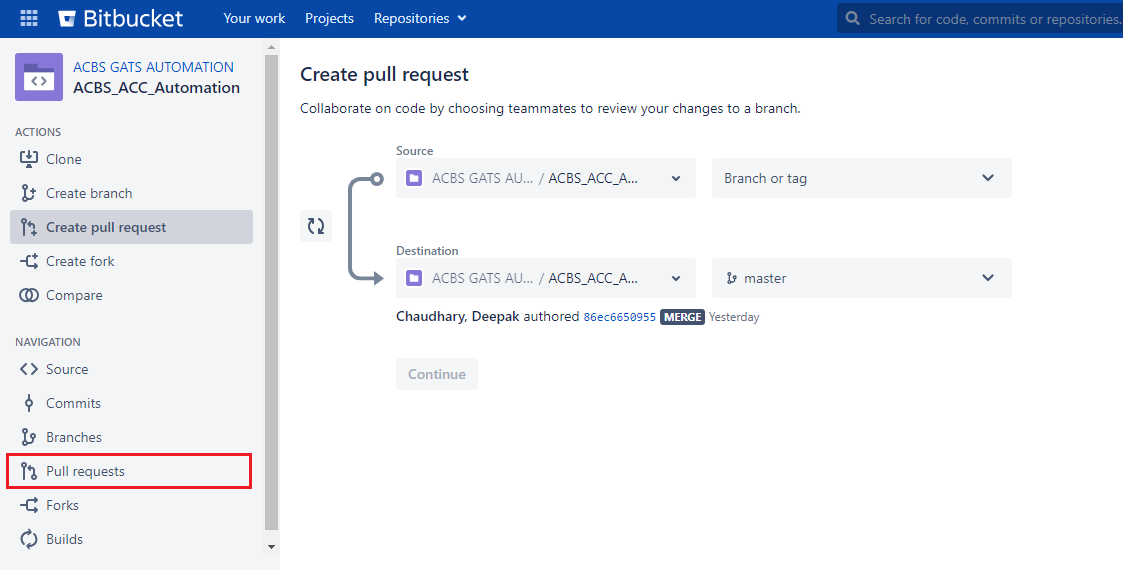
**git checkout branch\_name**

* Add your files/folder to your working directory where you have opened git bash
* git add .
* git commit -m "ticket\_name:description about commit "
* git push
* (After pushing one user WS, you may copy other user WS and replace files in that git working directory and follow these above 5 steps again)

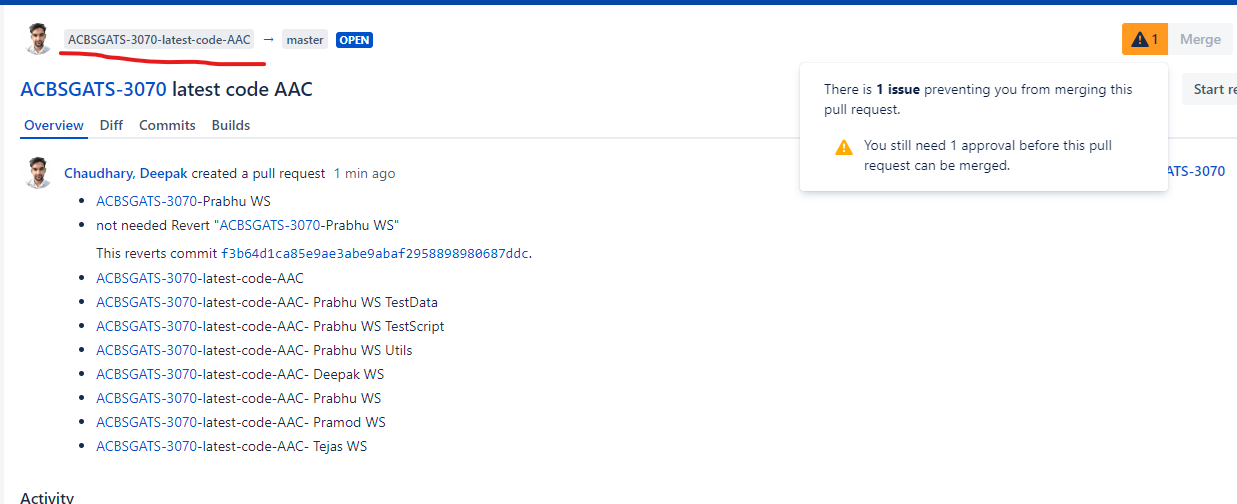
Now, your complete code is available in bitbucket, attached to branch created.



You have to create a pull request to merge this branch to master



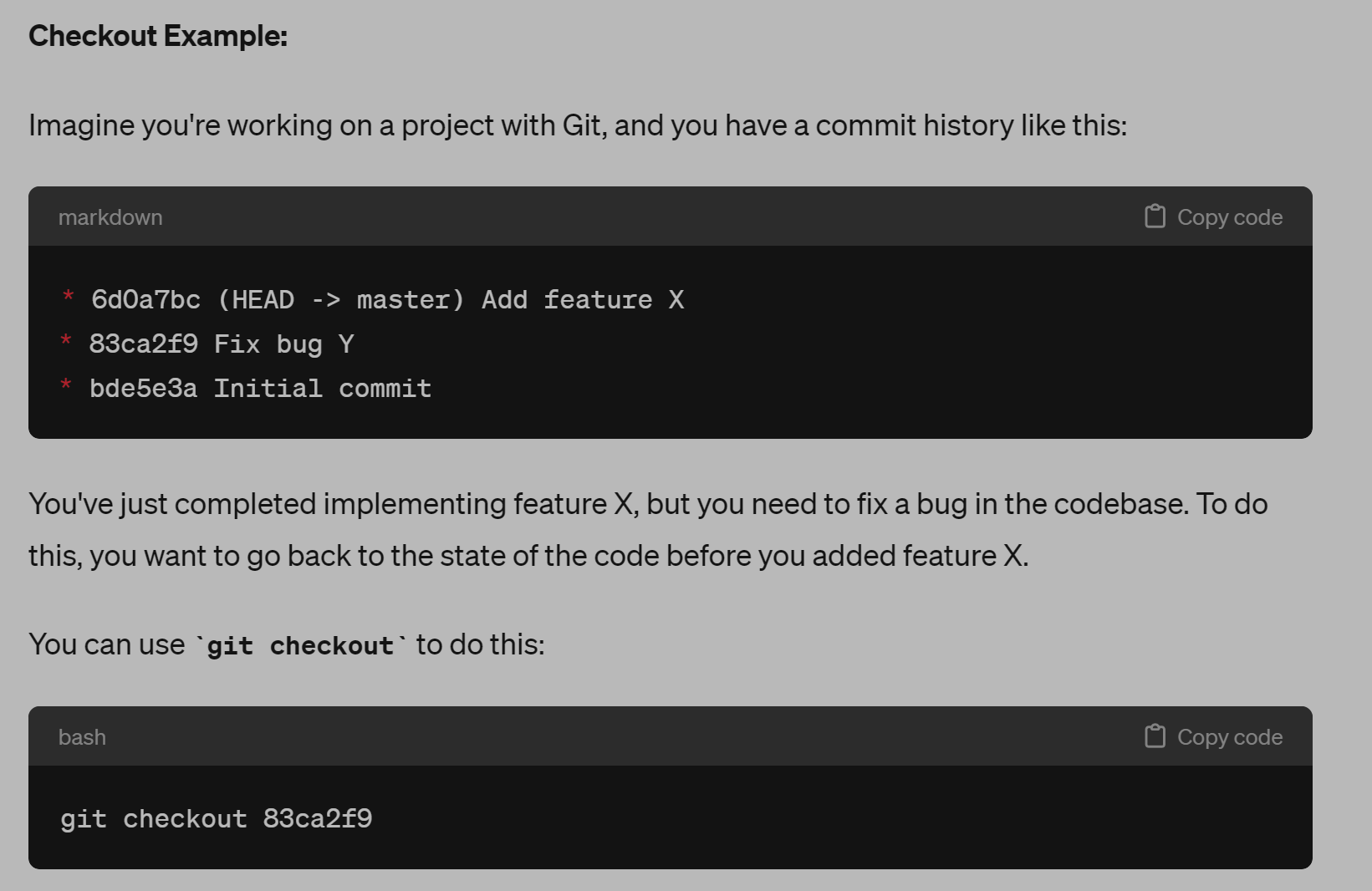
Once any team member approves that pull request, you can merge the branch to master.

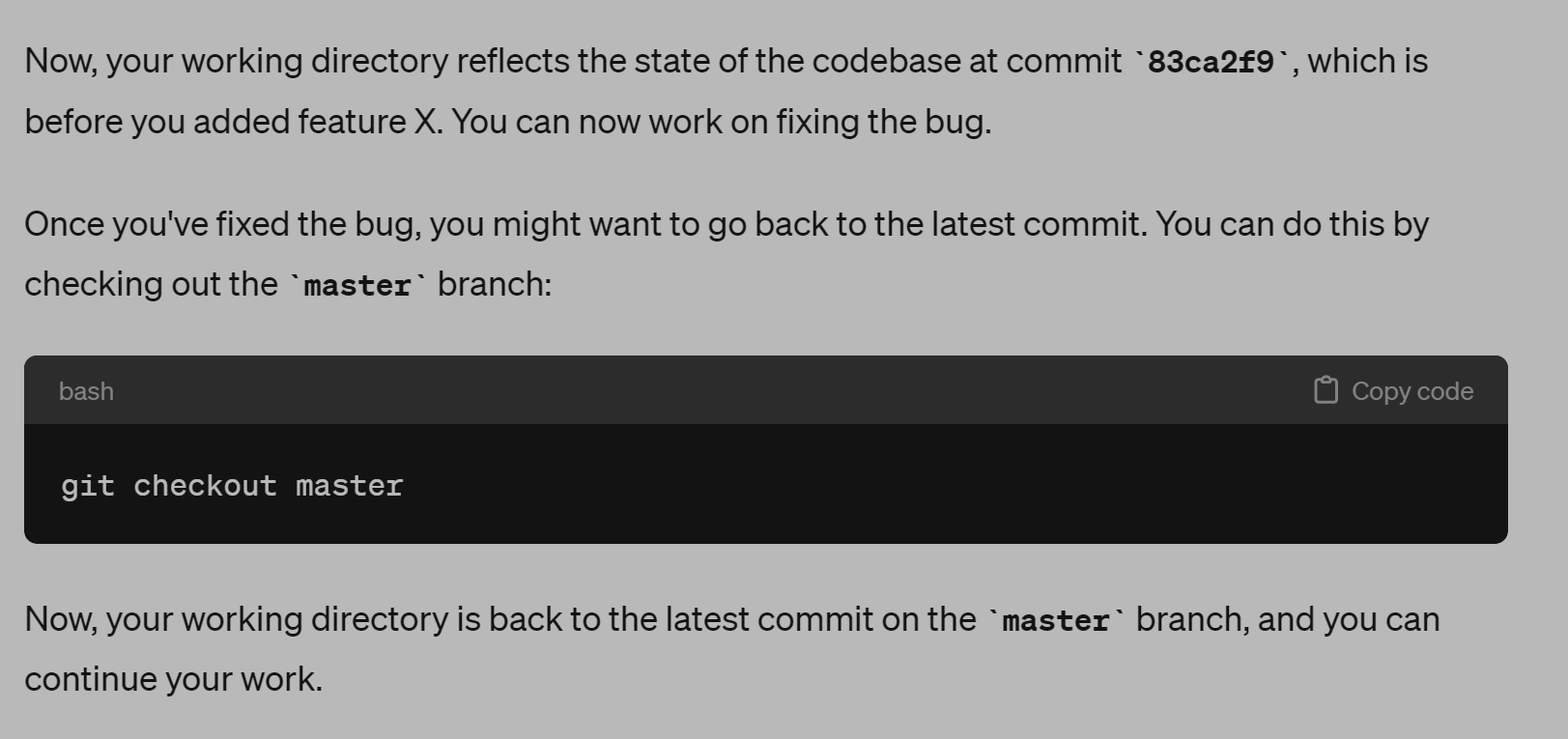


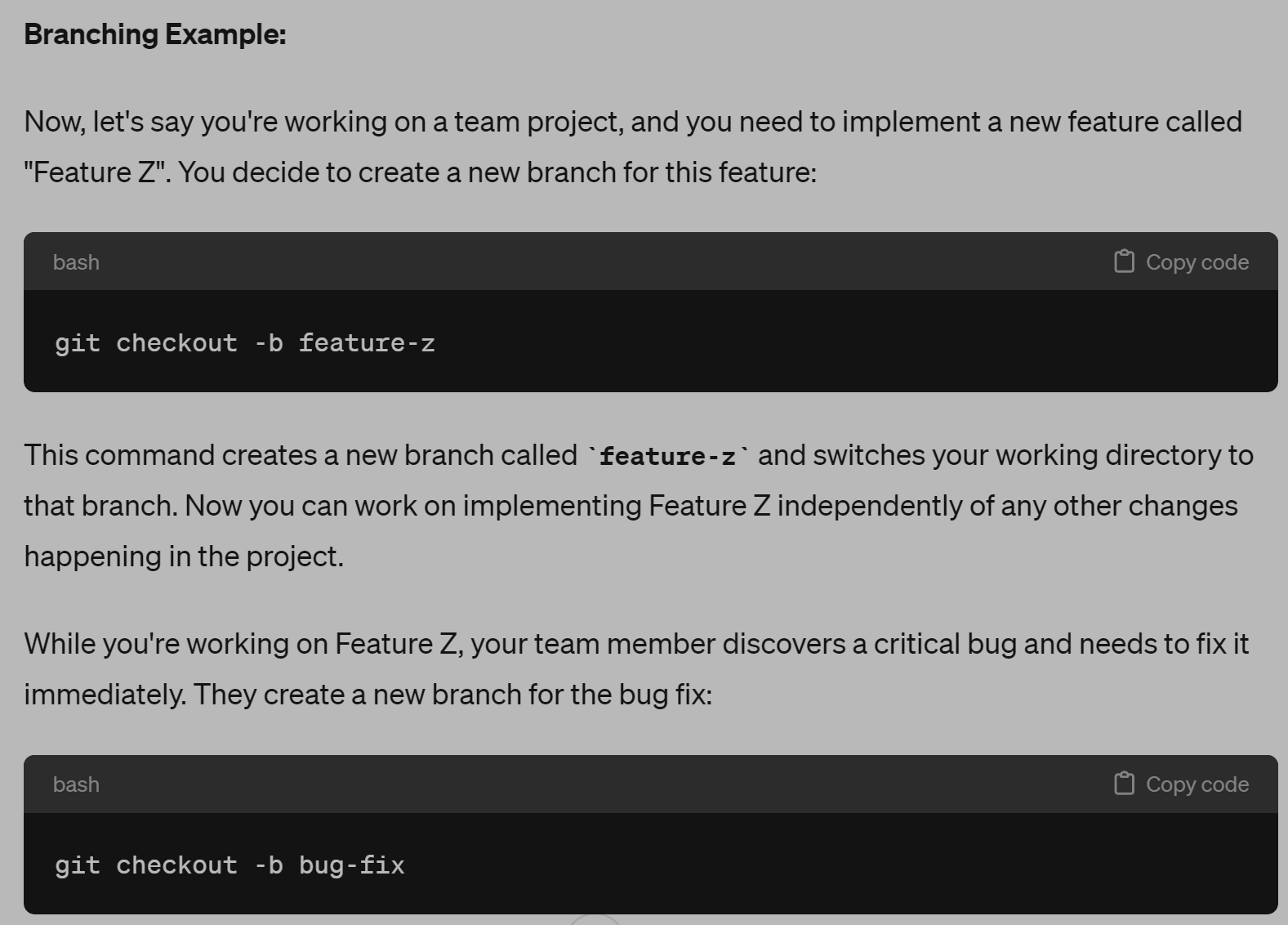
**Difference between Checkout and Branching**

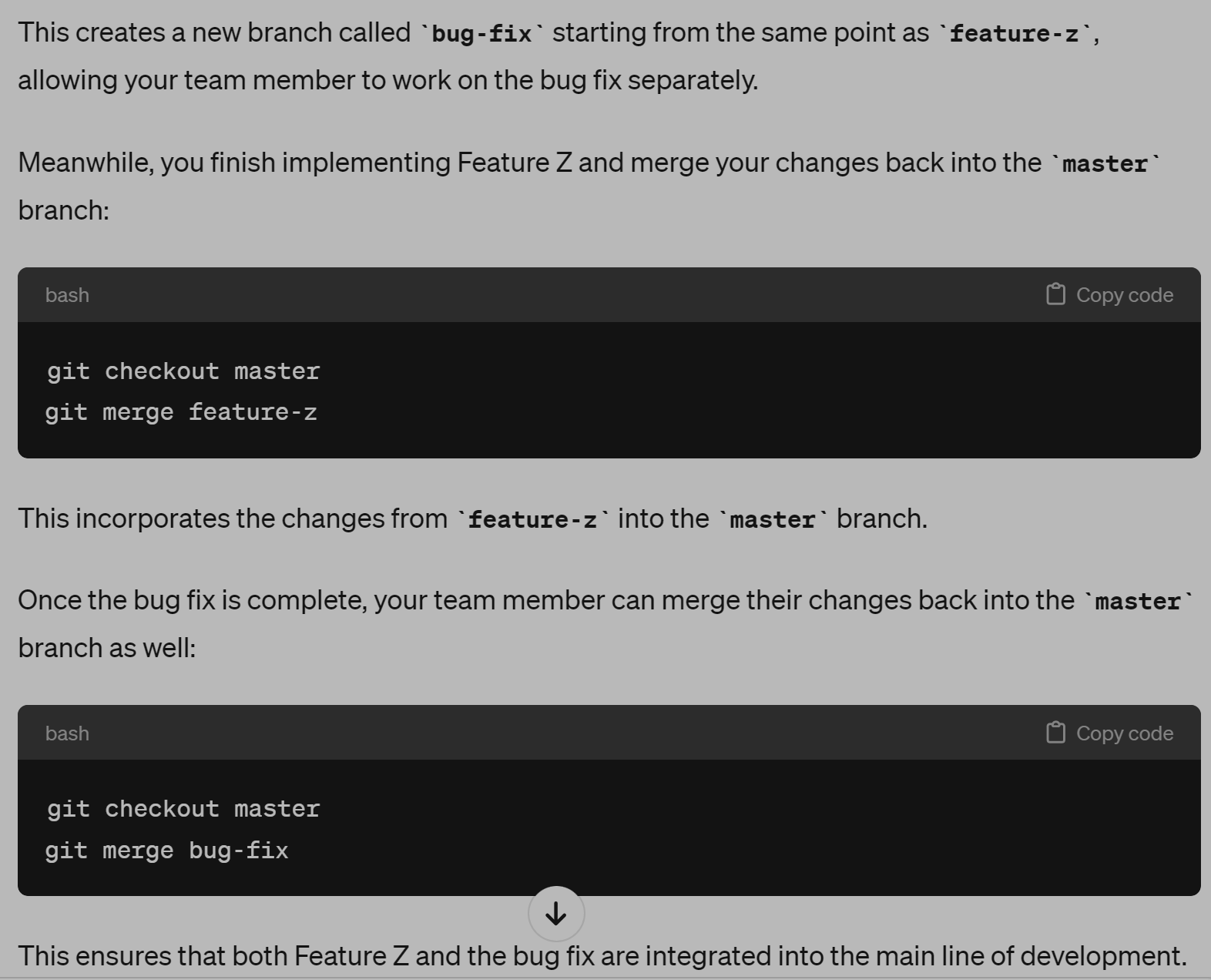
1. **Checkout**:
   * Checkout refers to the act of switching between different versions of a codebase.
   * It allows you to move your working directory to a specific state of the codebase, typically a specific commit or branch.
   * When you checkout a commit, you're essentially switching your working directory to reflect the state of the project at that particular commit. This can be useful for inspecting past states of the project, debugging, or reverting changes.
   * You can also checkout branches to switch between different lines of development in your project.
2. **Branching**:
   * Branching involves creating a divergent line of development from a specific point in the version history.
   * It allows multiple developers to work on different features or fixes simultaneously without interfering with each other's work.
   * When you create a branch, you're essentially creating a separate "timeline" of changes that can be merged back into the main line of development (usually the **master** branch) at a later stage.
   * Branches are lightweight and efficient in Git, making them a preferred method for managing parallel workstreams.

In summary, while both checkout and branching involve manipulating the state of a codebase, they serve different purposes: checkout is about navigating between different states of the codebase, while branching is about creating separate lines of development for parallel work.

****

****

****

****

**clear --- in git bash**

**cls ---in vs code**