😊 Code Reviews with Git Pull Request

A **Pull Request (PR)** in Git is a collaborative tool used in Git hosting platforms like GitHub, GitLab, or Bitbucket to request the merging of changes from one branch to another. Pull Requests are especially useful in the context of a **Git branching model**, as they facilitate code review, discussion, and quality assurance before changes are integrated into the main codebase.

Pull Request in Relation to Code Commits

1. **Definition**:
    - A Pull Request represents a set of changes (commits) that a developer has pushed to a branch and is now requesting to merge into another branch (usually main or develop).
2. **Workflow**:
    - A developer creates a new branch for a feature or bug fix (e.g., feature/login-page).
    - The developer commits changes to the branch as they progress.
    - Once the feature or fix is complete and ready for integration, a Pull Request is created.
    - The Pull Request highlights all commits made in the branch and shows the changes (diffs) compared to the target branch.
3. **Code Review**:
    - Other team members review the Pull Request to ensure the code is correct, adheres to standards, and doesn't introduce bugs.
    - Feedback can be given, and additional commits can be added to the branch in response to this feedback.
4. **Approval and Merging**:
    - Once the Pull Request is approved, it can be merged into the target branch.
    - The merge integrates all the commits from the feature branch into the target branch.
5. **Closing the PR**:
    - After the merge, the feature branch is typically deleted to keep the repository clean.

Pull Request Benefits

1. **Code Quality Assurance**:
    - PRs promote code reviews, helping identify potential issues or improvements.
2. **Collaboration**:

    - Team members can discuss implementation details and suggest changes directly within the PR.
3. **Traceability**:
    - Every Pull Request serves as a record of what changes were made, why, and by whom.
4. **Branch Isolation**:
    - Developers can work independently on features while the main branch remains stable.
5. **Continuous Integration (CI)**:
    - Many teams integrate automated testing tools with PRs, ensuring that the code passes tests before merging.

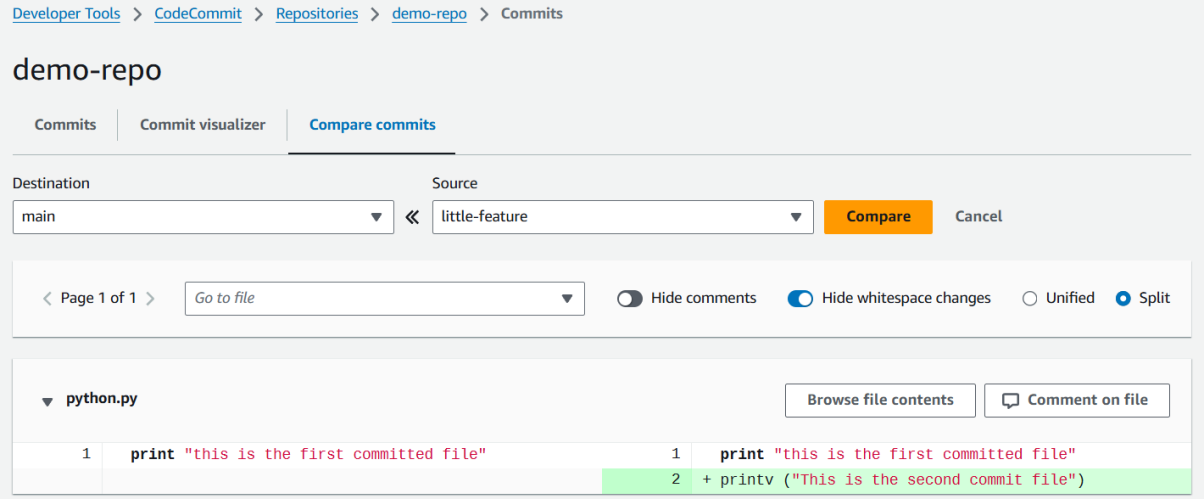
Pull Requests and Commits Relationship

- **Granularity**:
    - A PR aggregates all commits from a branch, making it easier to track related changes as a single unit.
- **Amendments**:
    - Developers can continue to add commits to the branch after opening a PR. These new commits are automatically reflected in the PR.
- **Atomic Changes**:
    - Encouraging smaller, focused commits helps maintain clarity in PRs.

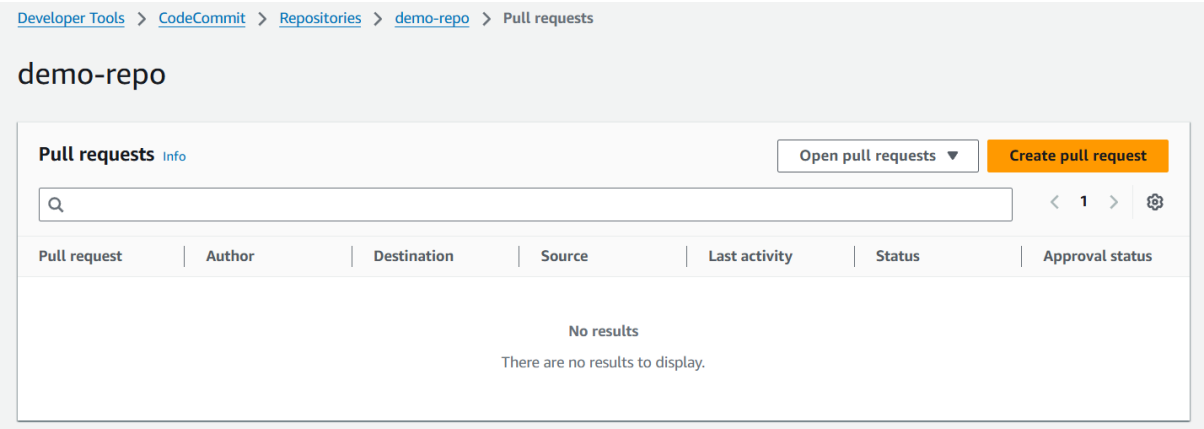
By integrating Pull Requests into the Git branching workflow, teams can ensure that new code is rigorously reviewed, tested, and seamlessly integrated into the main codebase.

😄 To begin with the Lab:

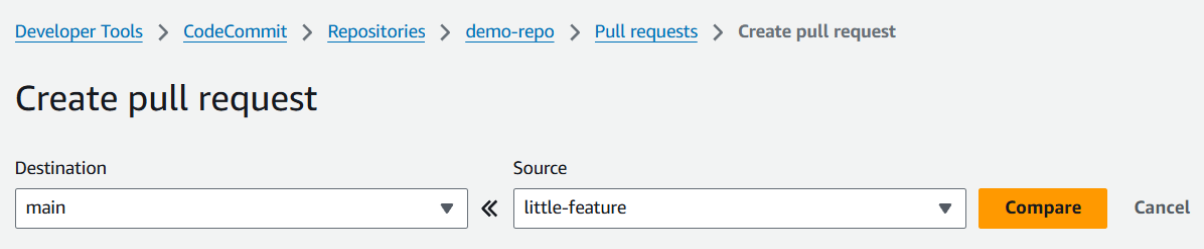
1. In this lab we are going to learn about the Git pull feature available in Code commit.
2. Pull requests are generally used in conjunction with the Git branching workflow. In simple terms, pull requests are a mechanism for a developer to notify team members that they have completed a feature. This lets everybody involved know that they need to review the code and merge it into the master branch.
3. Now we were working with the git branching in our last lab and on the right side we have a little-feature branch and on the left, we have the main or master branch.
4. Here we know that our main or master branch is working perfectly and the developer has notified us that the little-feature branch is working perfectly and we can merge it with the main branch. So, what will happen the code from the right side will go to the left side or the production environment.



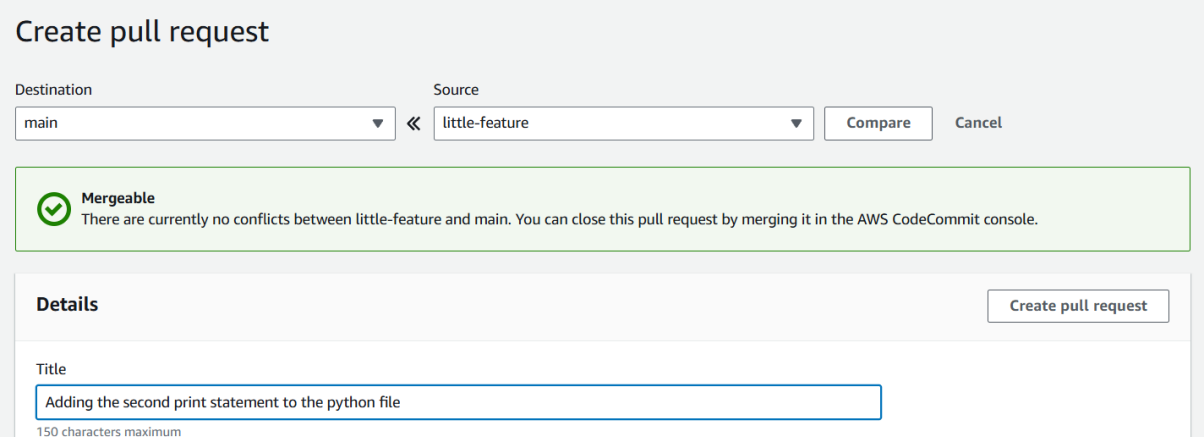
5. In order to do that we have a tab for pull requests on the left pane. We can go to pull the request and create it.



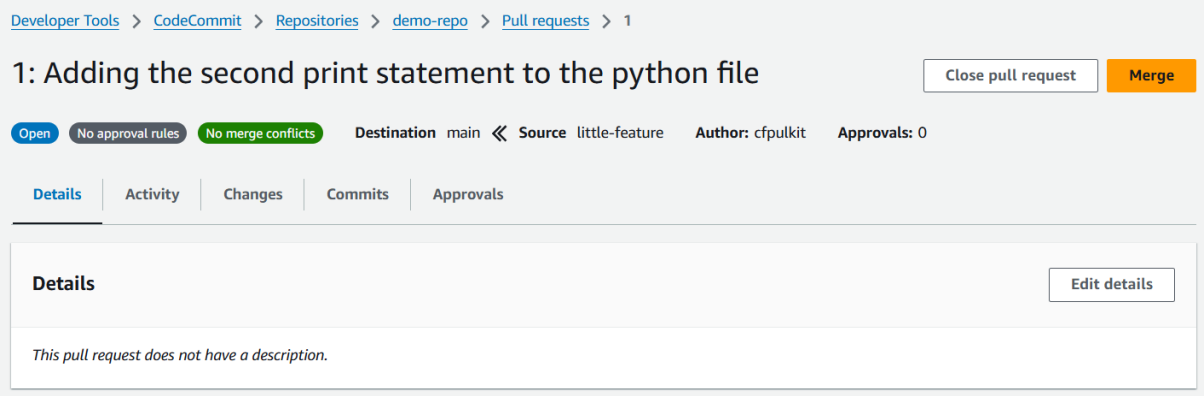
6. Now our destination should be the main branch and the source should be the little-feature branch. Then click on Compare.



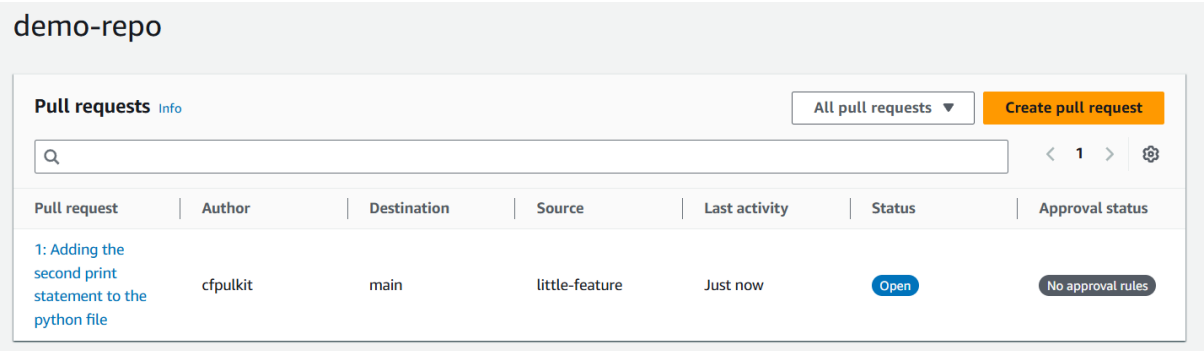
7. Here you can see that there are no conflicts and the content can be merged. So, we need to give it a title and click on create a pull request.



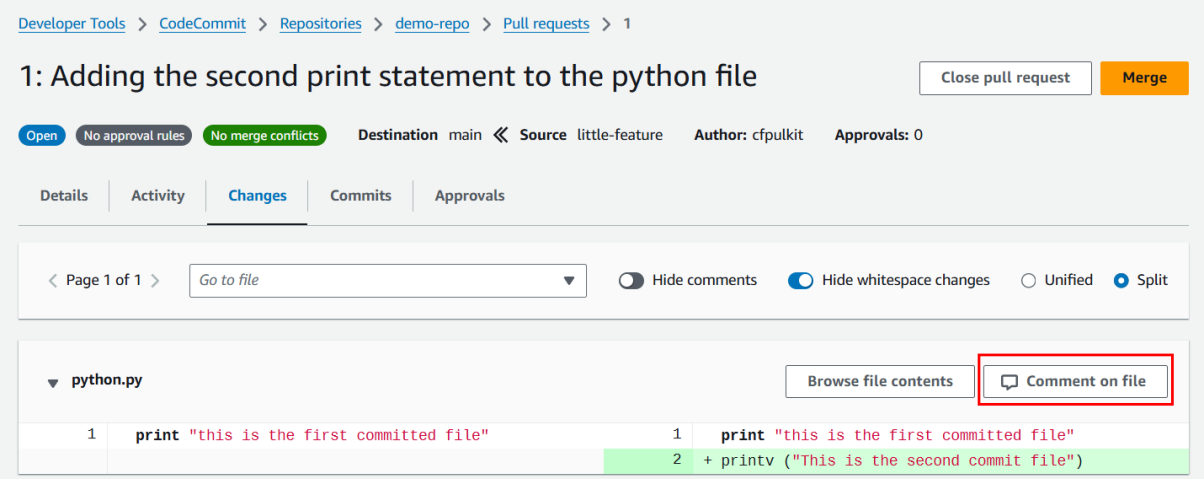
8. This is how the pull request would look like. Now we can go back.



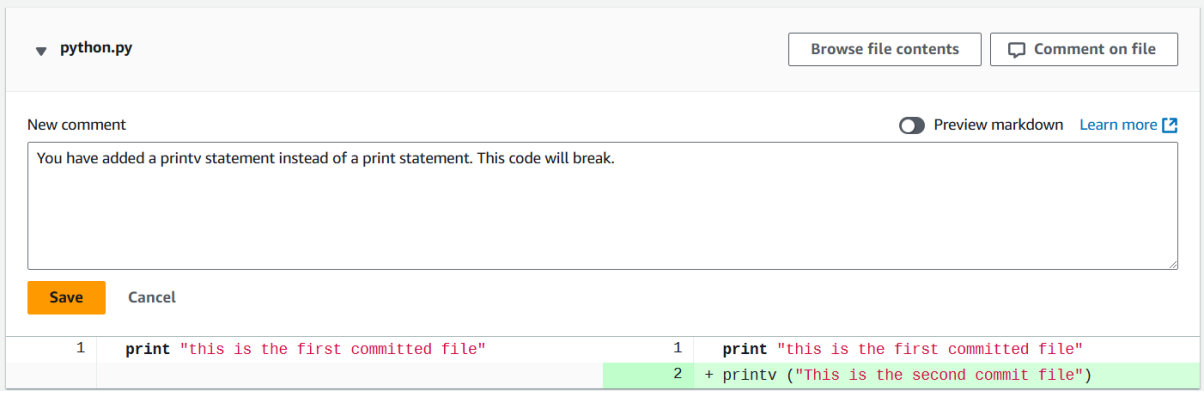
9. As you can see here the status for the pull request is still open which means that it has not been merged yet. This means that once the developer has completed his work he will generate a pull request and this request will reviewed by multiple team members before it can be merged with the master or main branch.



10. So, if we go inside of it, we will be able to see the appropriate changes. We can also comment on the changes using the comment-on-file feature.



11. So, if you see here there is an error with the print statement that is in the second line we have created a typo of printv instead of print.
12. Based on this we are adding it in the new comment. Click on Save.



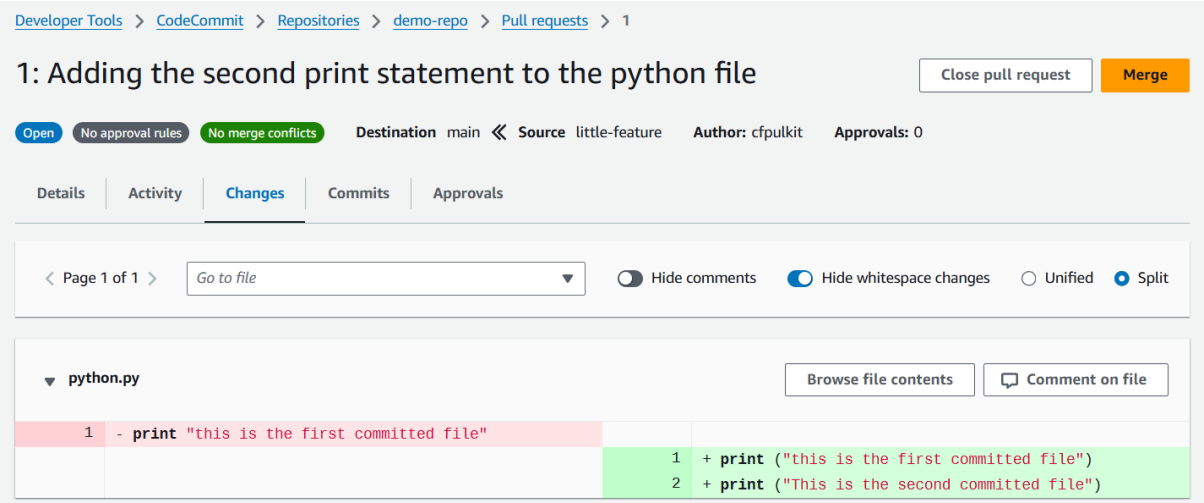
13. Also, if you run the python.py file then you will see that there is an error with the 2nd line.

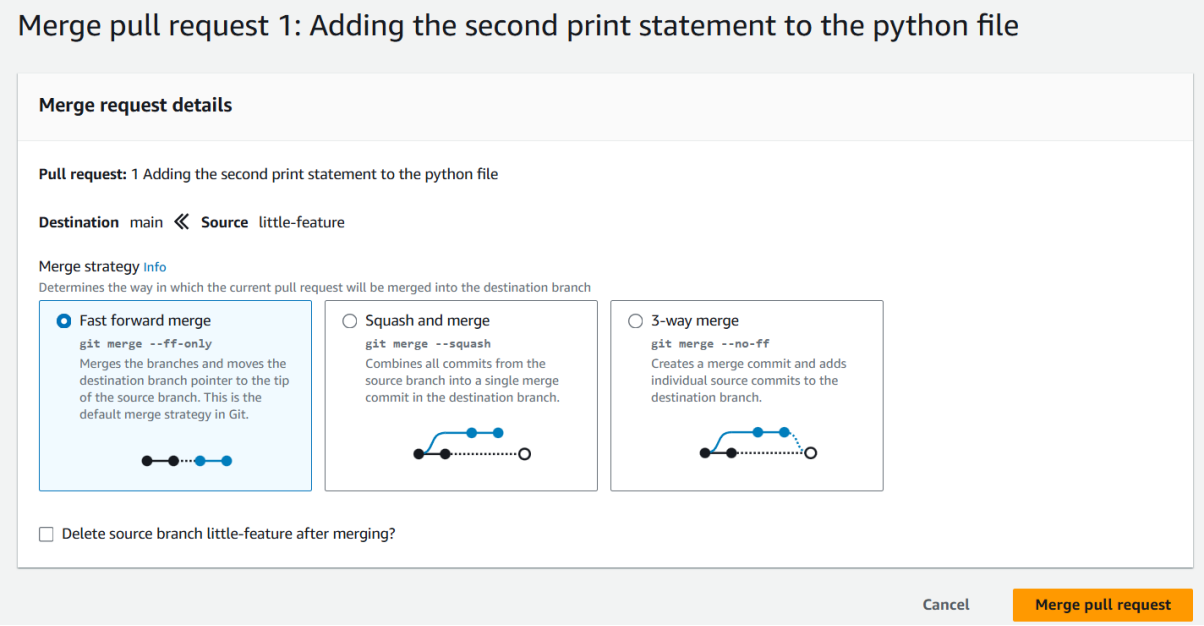
```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/tomcat/demo-repo (little-feature)
$ python python.py
this is the first committed file
Traceback (most recent call last):
  File "C:\Users\PULKIT\Desktop\tomcat\demo-repo\python.py", line 2, in <module>
    printv ("This is the second commit file")
    ^^^^^^
NameError: name 'printv' is not defined. Did you mean: 'print'?
```

14. By this the developer will realize that he has made a mistake then he will correct this and push the changes to the repository.

```
      @LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/tomcat/demo-repo (little-feature)
$ nano python.py

      @LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/tomcat/demo-repo (little-feature)
$ git add .

      @LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/tomcat/demo-repo (little-feature)
$ git commit -m "Fixing the printv statement typo"
[little-feature 496d905] Fixing the printv statement typo
 1 file changed, 2 insertions(+), 2 deletions(-)

      @LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/tomcat/demo-repo (little-feature)
$ git push origin little-feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Validating objects: 100%
To https://git-codecommit.ap-south-1.amazonaws.com/v1/repos/demo-repo
   c0b6c99..496d905  little-feature -> little-feature
```

15. Once the changes have been pushed you will see that the changes have been made. Now can click on the Merge button to





16. If you go to your code file and open it you will see that it has been updated.