**Integration between Jenkins + JIRA + bitbucket servers and Implement CI/CD**

|  |
| --- |
| Note: Please view this image in “Web Layout” to see the complete image,  right side bottom of the word doc. |

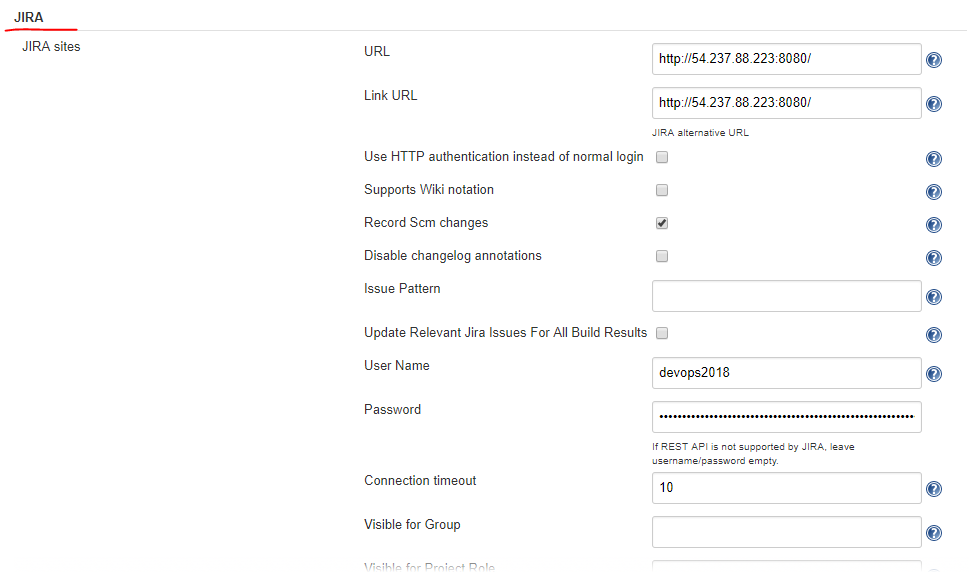
**Jenkins + JIRA:**

**Install “JIRA plugin” in Jenkins:**

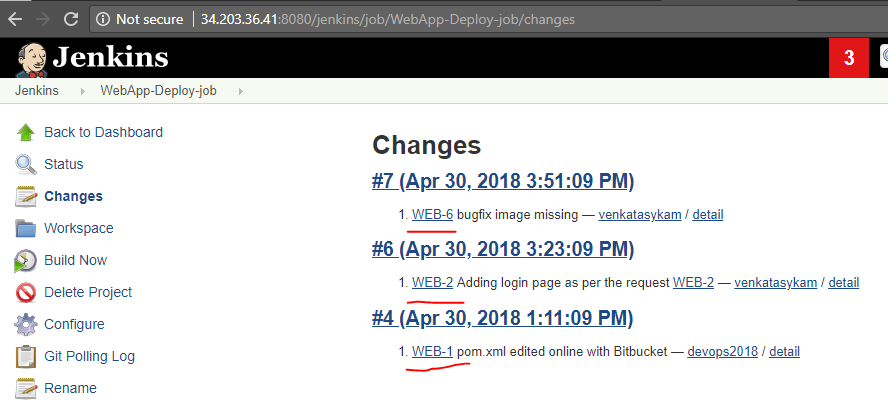
Go to “Manage Jenkins” 🡪 “Manage Plugins” 🡪 Available



Go to “Manage Jenkins” 🡪 “Configure System” 🡪 Add the JIRA URLs & credentials.

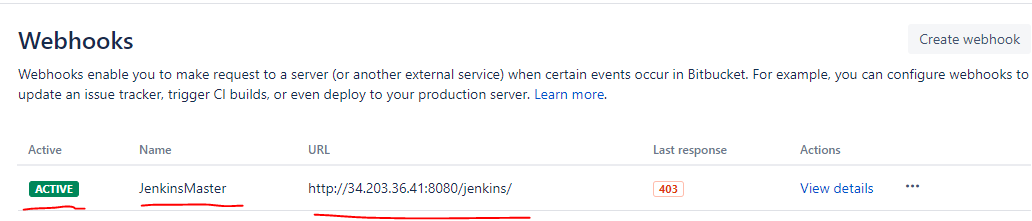


You will be able to see the link to the JIRA issues for every Jenkins job changes section. Click on the JIRA issue, and then you can directly navigate to JIRA issue from here. Easy to track the each & change “Who & why the change made in the code repo”.



**Bitbucket + Jenkins**: Once the changes are done in bitbucket repo, then the Jenkins job will be triggered automatically (based on the ‘Poll SCM’ config).

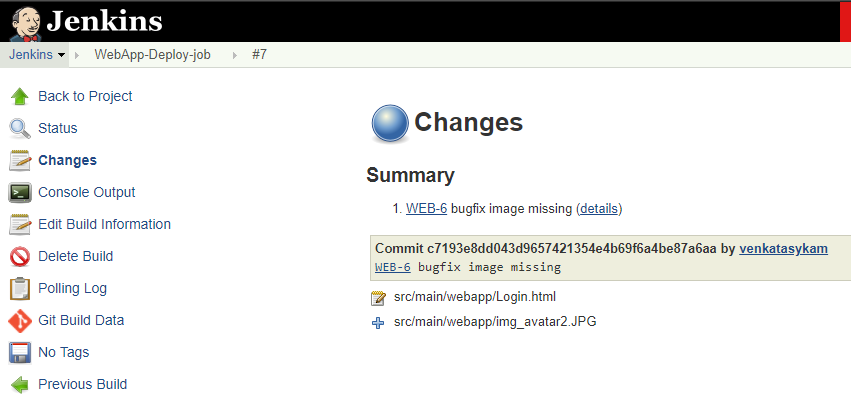
Bitbucket repo or project settings🡪 Webhooks 🡪 Create a webhook 🡪 give name & URL & checkbox “push”.



Configure each job with Poll SCM as to trigger the build automatically after 1 min the changes made on the branch.



Click on changes 🡪 details🡪 you will find the commit id along with files which are changed as part of this commit.

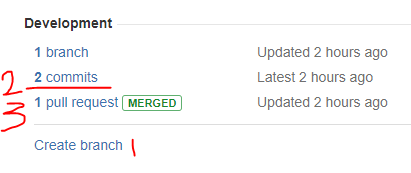


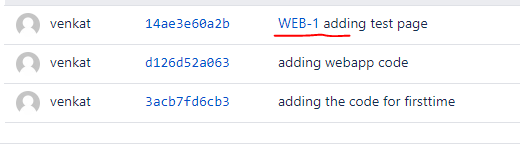
**Bitbucket + JIRA**:

Go to Bitbucket settings 🡪 Application Links 🡪 Enter the JIRA URL & Click on Create new Link 🡪 it will navigate to JIRA and prompt you to enter the creds to finish the integration.

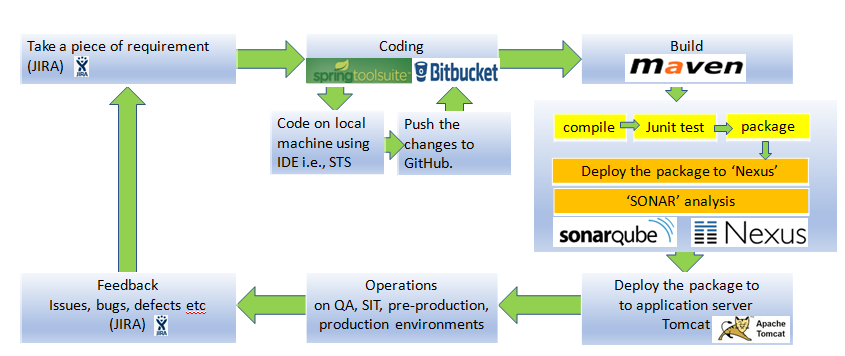
This setup will provide us some features as below.

1. Create branch/pullRequest from JIRA issue.
2. Each commit tracked by the JIRA ticket.
3. Pull request status in JIRA issue.
4. Clickable link as to directly navigate to the JIRA issue in the bitbucket commit history (commit message). i.e., smart commit.





Establish a communication between the servers Jenkins, JIRA , bitbucket to implement CI/CD as shown in the below image. (Skipping the sonar analysis as this is done in Phase-3 & Nexus package deploy as this is done in Phasae-1).



Here is the basic scenario to define the DevOps lifecycle. i.e., CI/CD.

**Step-1:** Divide the requirement into small pieces for each developer. WEB-2, WEB-3, WEB-4.

**Step-2:** Do the changes in local with STS or Notepad editor.

**Step-3:** Commit the changes & push to bitbucket repo.

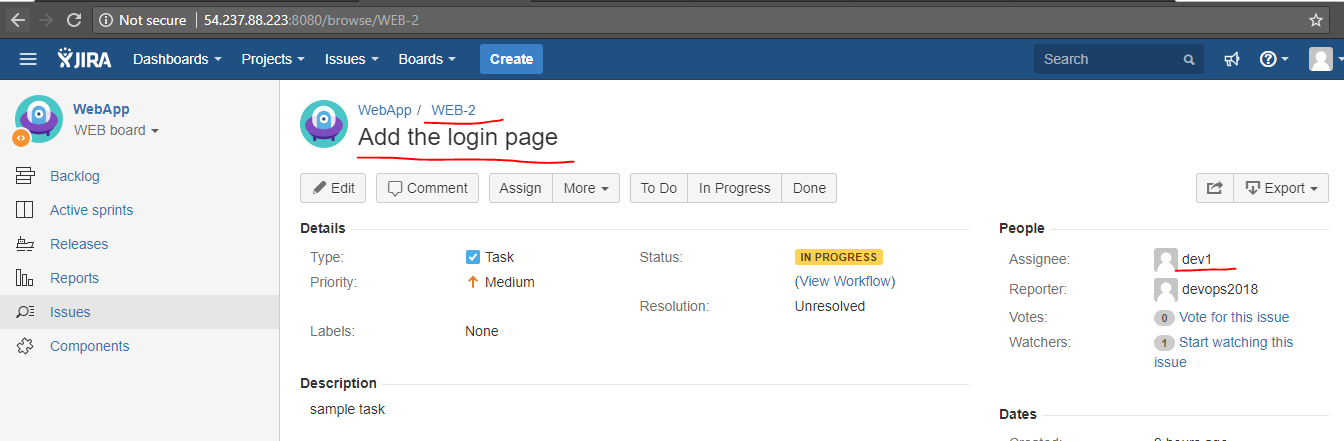
**Step-4:** Create pull request for review purpose.

**Step-5:** Once the PR closed (merged), auto trigger the Jenkins job as to build with new changes and deploy the latest package on Tomcat server.

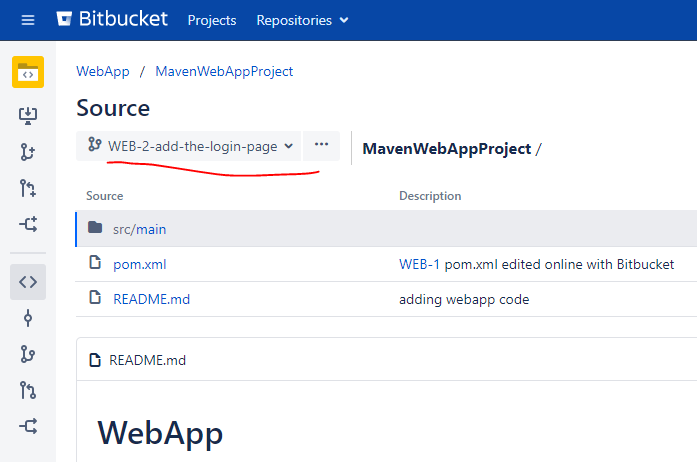
**Step-6:** Do the changes again if there are any additional changes required to meet the quality of the app. Repeat Step-1.

**Step-1:** Divide the requirement into small pieces for each developer. WEB-2, WEB-3, WEB-4.

**Developer1 is Working on WEB-2 task:** Add login page (dev1 is going to work on it), see the ticket is in-progress & dev1 owned the ticket. Dev1 user is going to start creating branch & make some changes as per the requirement WEB-2.



Branch created: Click on “Create Branch” under the Development.--> Click on “Create”.

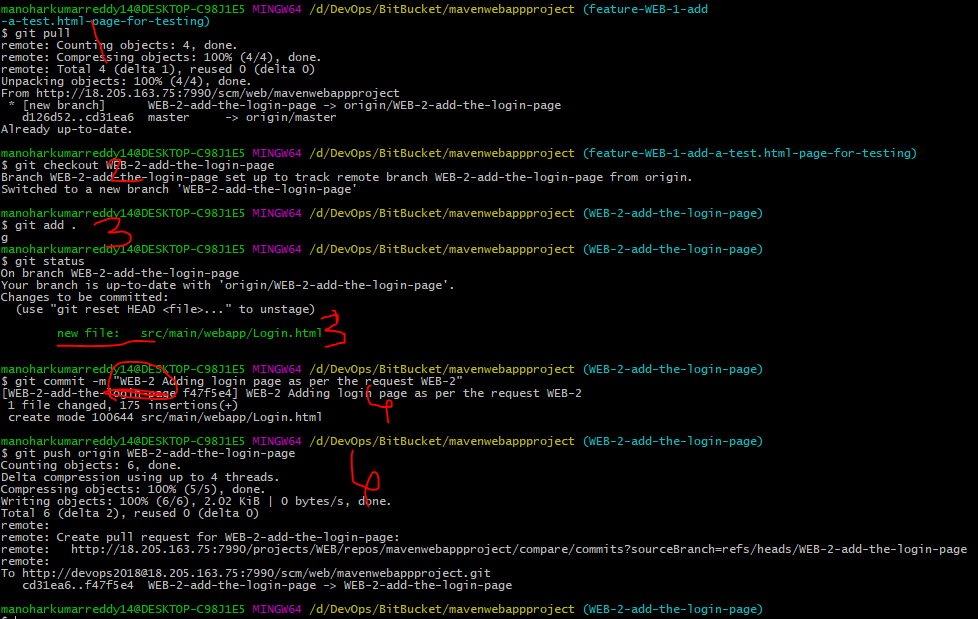


Now, **Dev1** user is going to switch to the new branch & making changes as per the requirement WEB-2.

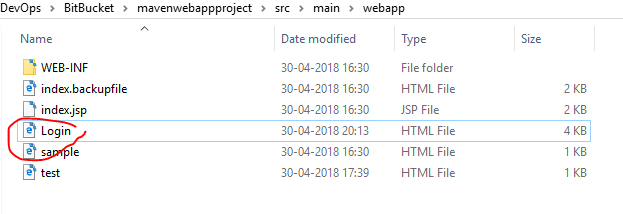
For this,

1. First pull/update the local workspace: git pull
2. switch to new branch: git switch <new branch name>
3. make the required changes: adding the **login.html** page : git add . or git add Login.html (check the status what are all the files added & ready for commit)
4. and then commit & push : git commit -m “<commit message with JIRA ticket reference, see below image with rounded one>”.

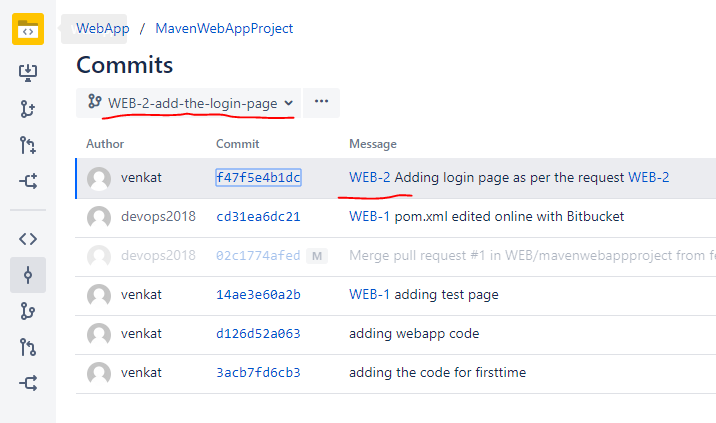
Please see the below image with the above point reference numbers.



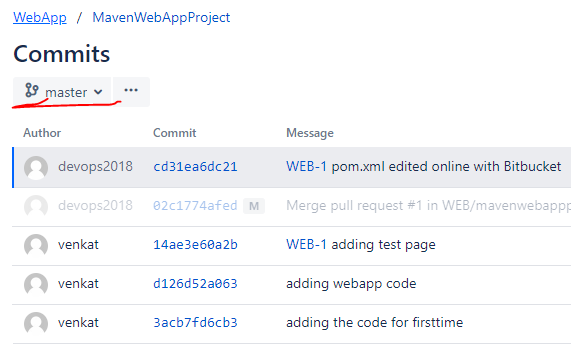
Added a new file locally:



Commit history on the new branch: New changes pushed to new branch (NOT the master).



Commit history on the master: There is no NEW commit/change on master branch.



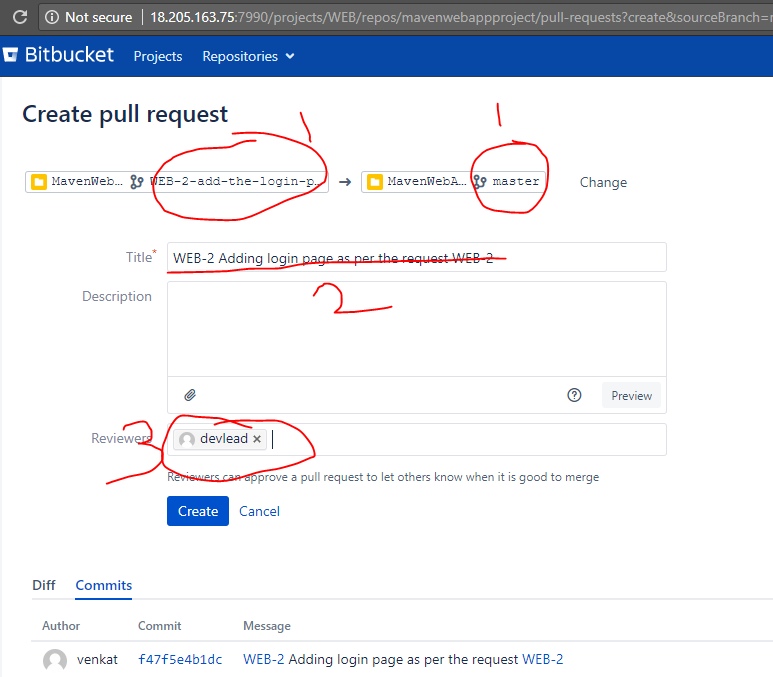
Since the Jenkins branch & the production code is on the ‘master’ branch(assume this is our development strategy) we need to merge the Dev1 changes from new branch to the branch ‘master’. So that Jenkins job will be triggered on the ‘master’ branch and jenkins will deploy the latest code on tomcat server, so that we can proceed with further testing.

For this, lets create the PR (Pull Request) by Developer1 and then, reviewer/devlead will review the changes done by the dev1 user and merge the changes to ‘master’ branch from new branch if everything looks good.

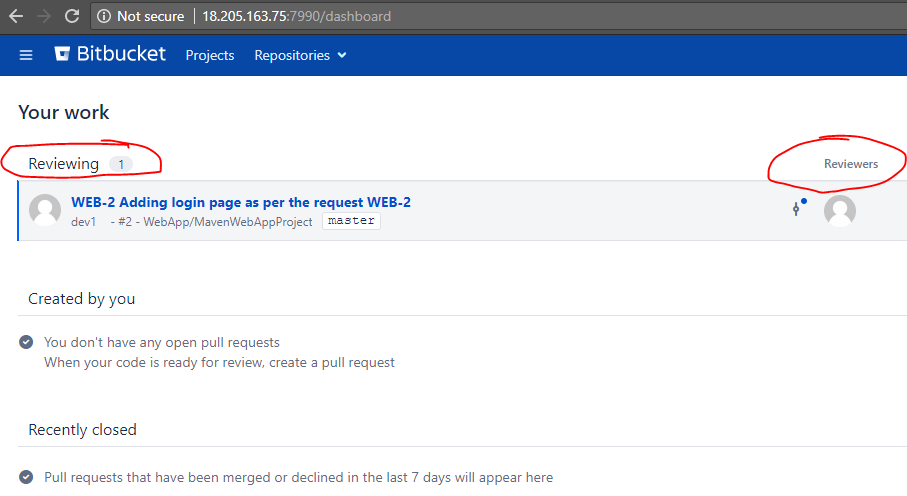
Go to JIRA issue WEB-2-🡪 Click on branch 🡪 Click on ‘Create Pull Request’🡪 next see the below image with the below point referring the number.

1. Creating pull request for merging the changes from new branch to master branch.
2. Pull Request(PR) is for merging the changes from new branch to master branch, the changes/commit made by the user Dev1.
3. Assigning this PR to **devlead** to review this request.

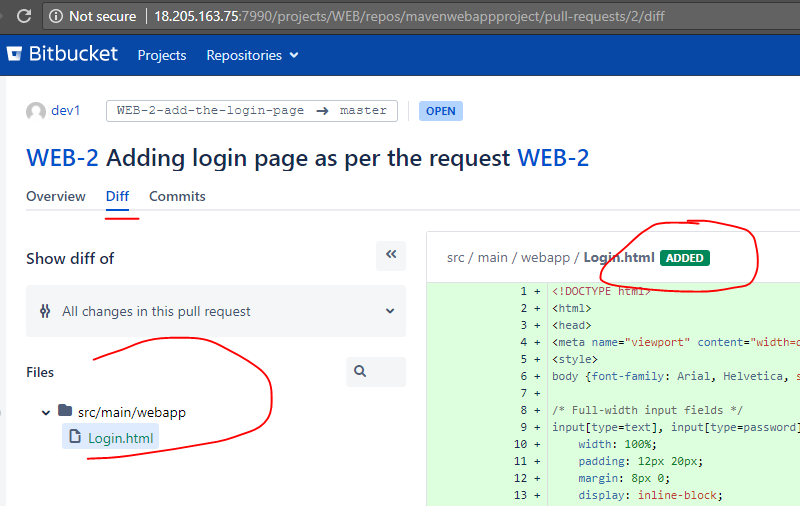
Click on ‘Create’ to create the PR.



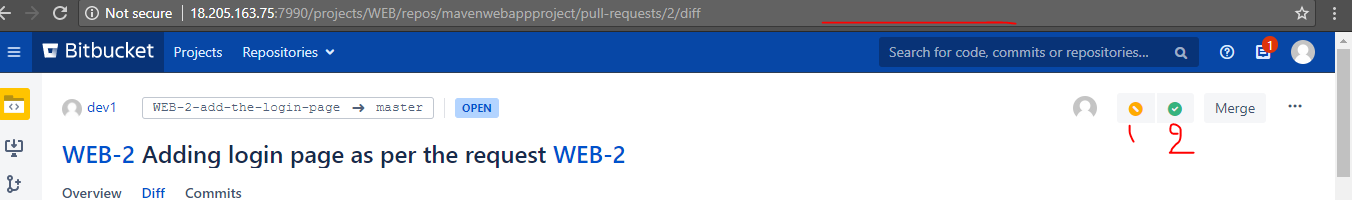
Once the ‘devlead’ login into the bitbucket, user can able to see the PR as pending for review.



* Click on the PR and review the changes & proceed with merge if everything file. In this review, reviewer may notice that, one file ADDED.

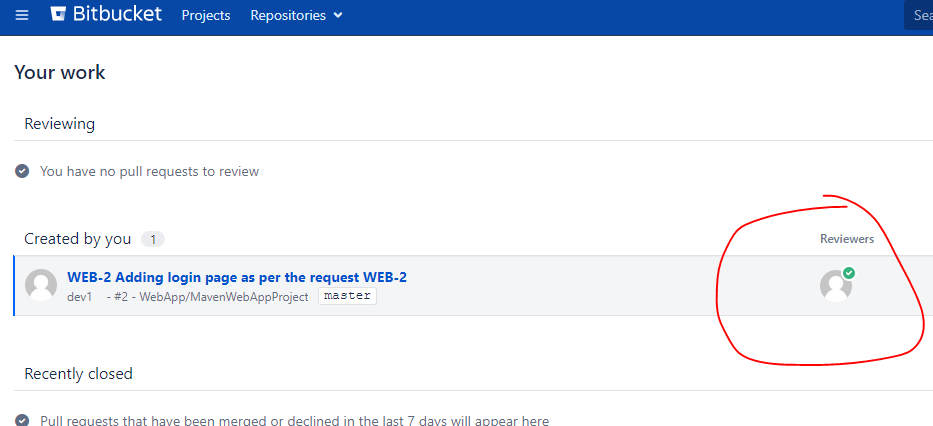


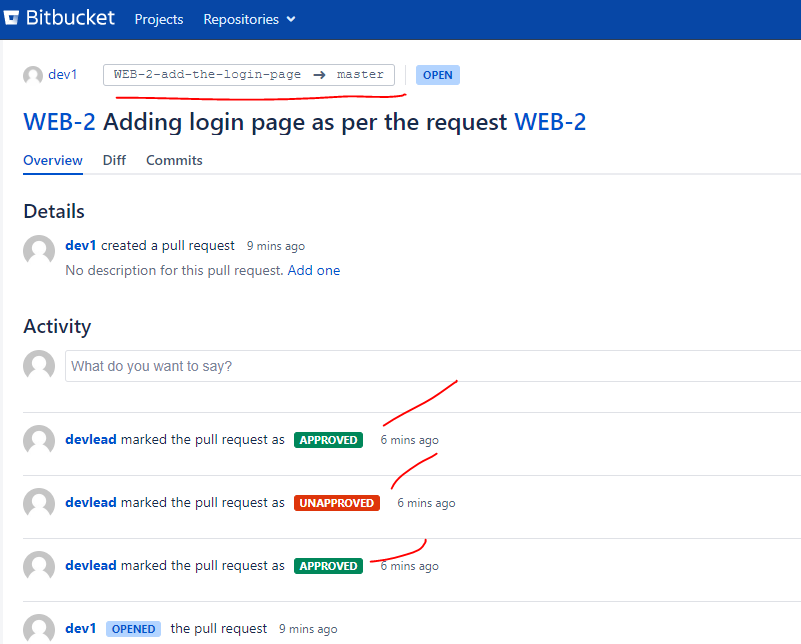
1. Deny & 2. Approve this PR.



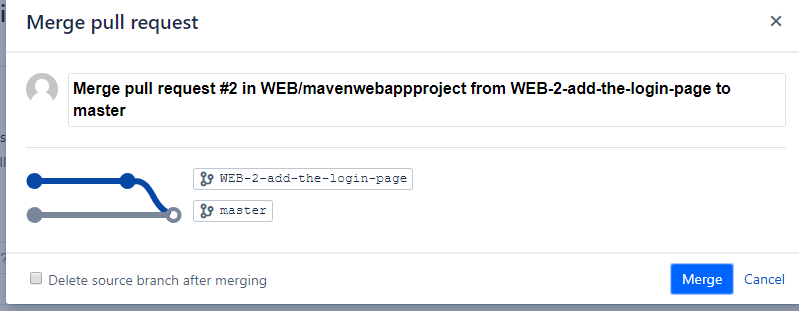
Once the PR, approved by reviewer, ‘dev1’ user can proceed with the merge to ‘master’ branch.

See the green ticket once the review success.

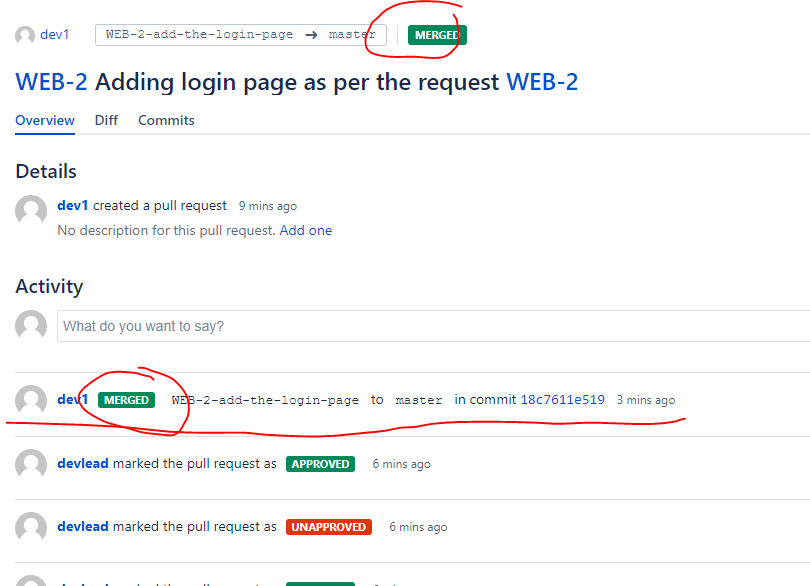




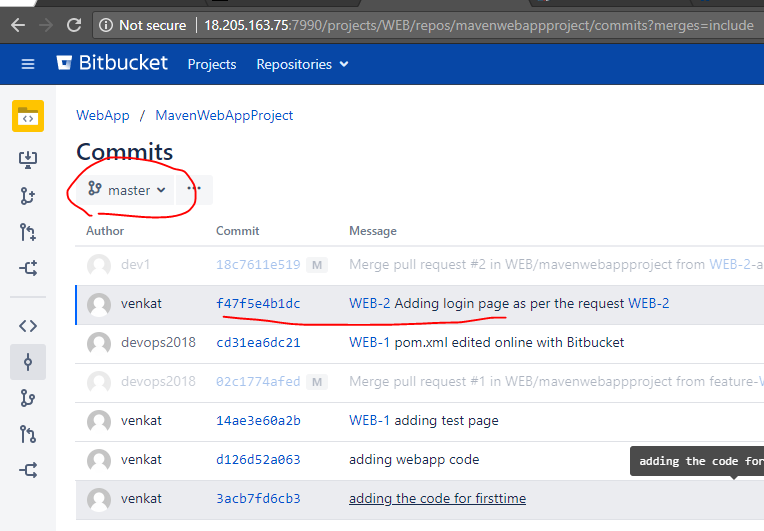
Click on ‘merge’.



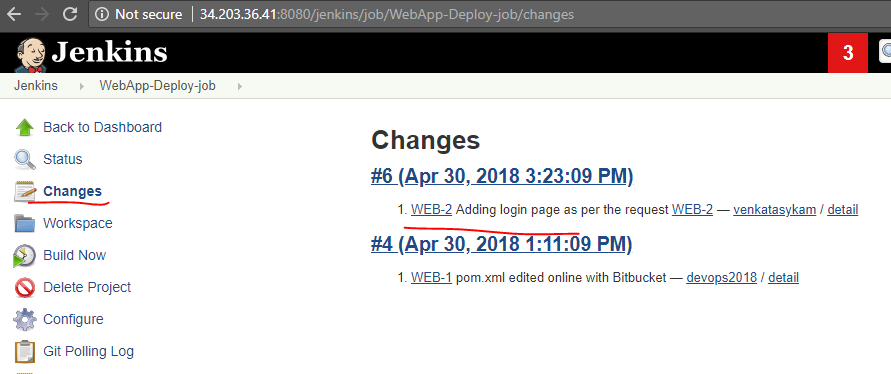
Merge success: See the activity log.

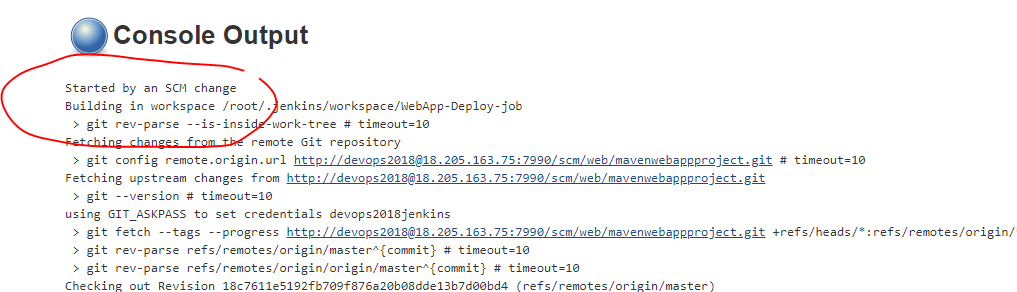


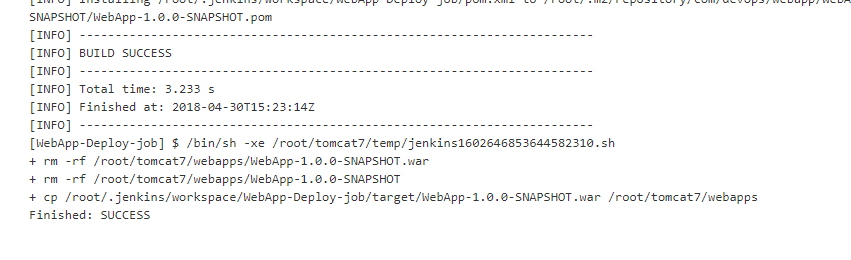
See the commit history on master branch after the merge.



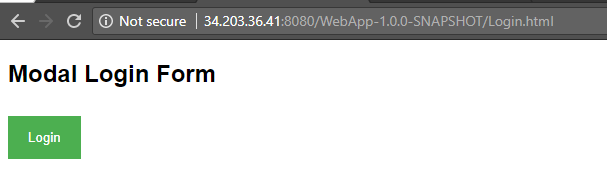
Now go to Jenkins job: Job triggered automatically by SCM changes.





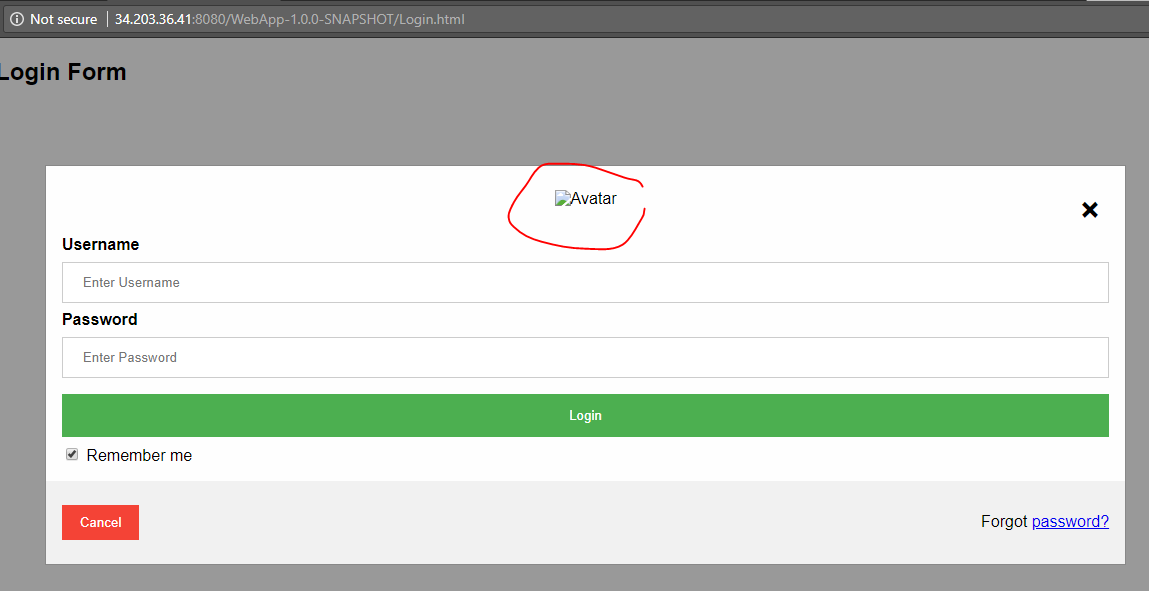


<http://34.203.36.41:8080/WebApp-1.0.0-SNAPSHOT/Login.html>

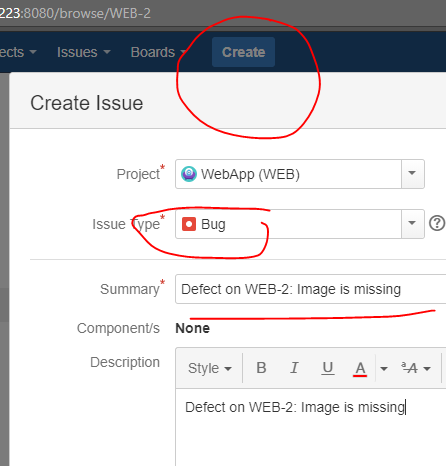


Now let’s proceed with testing, click on **Login** button & see the Login form whether the output is expected format or not.

Just click on the button Login, the Login form is displaying the image ‘Avatar’ as shown below. This is bug/defect. Tester will open a defect/bug to dev team as the image is not loading.



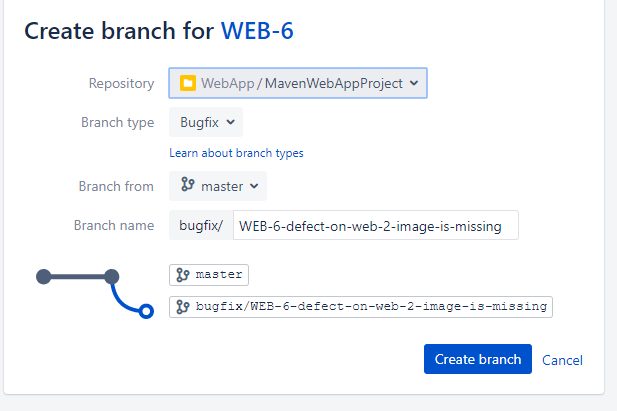
**Create a Bug:** Go to JIRA 🡪 Click on Create🡪 Choose issue type as ‘Bug’ give the proper summary & description and the create the Bug.



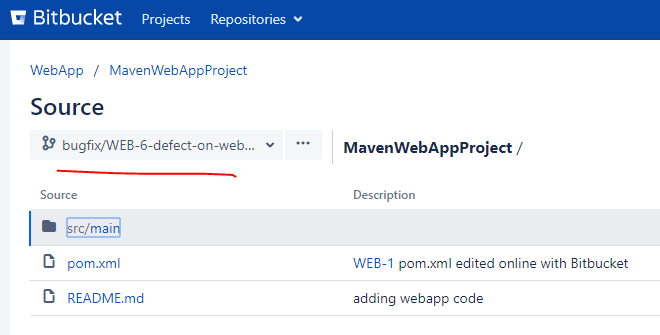
Tester created the issue as bug in JIRA project and assigned to dev1 or dev team.

Dev1 / Dev team will own this JIRA issue/bug & start finxing it. Dev1 user repeated the same steps 🡪 Create a feature branch 🡪 git pull localy(i.e., update the local workspace) 🡪 git switch to new branch 🡪 update the required changes as per JIRA bug which is opened by the tester for Dev team 🡪 push to remote 🡪 create PR for review 🡪 reviewer will appove or deny based on the bug details 🡪 Dev1 will merge if it is approved , dev1 will do the chanegs again if PR is denied. 🡪 once the merge is done, Jenkins deploy job will be triggered automatically. See the images below for all these steps one by one.

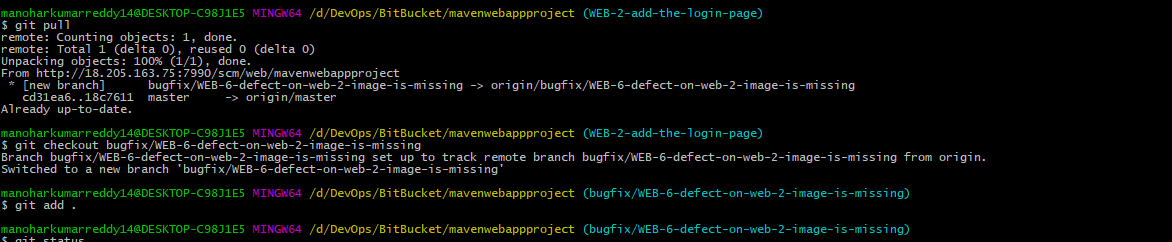
Click on “Create branch” 🡪 Development secion in JIRA issue.

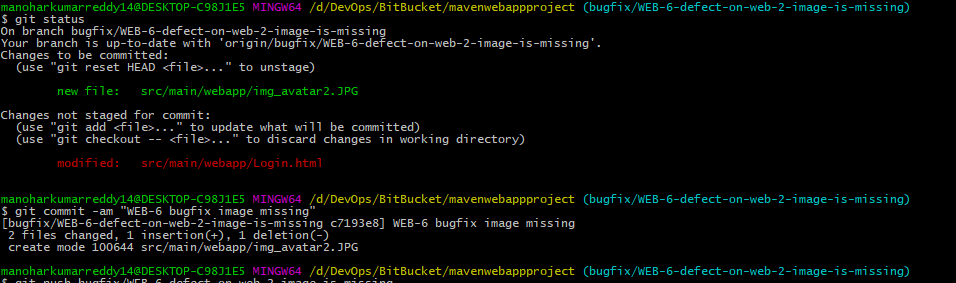


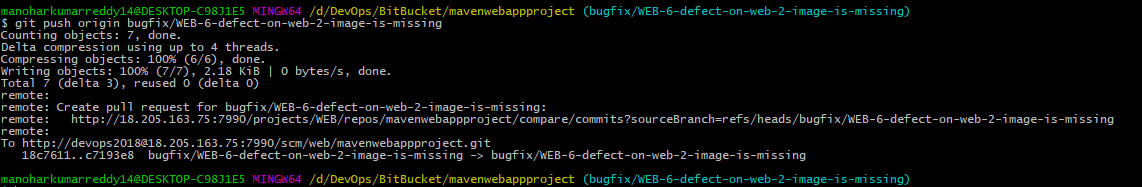
New branch created for bugfix:



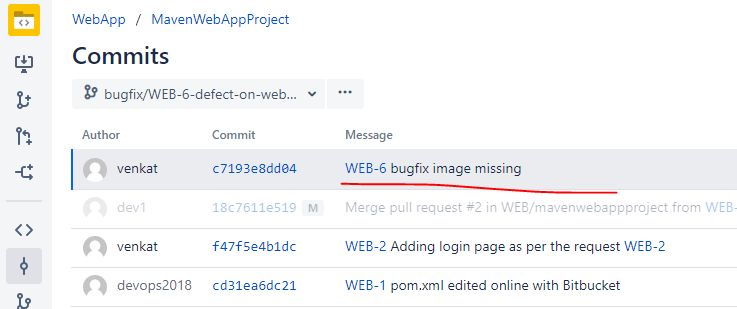
Updating & commiting the changes with JIRA issue:



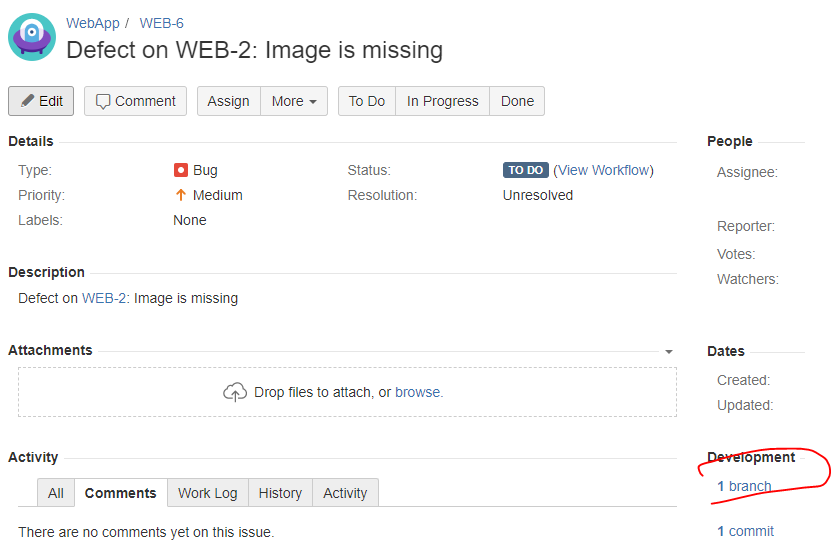


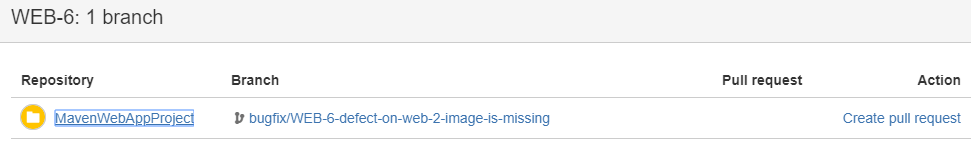


Commit history on new branch:

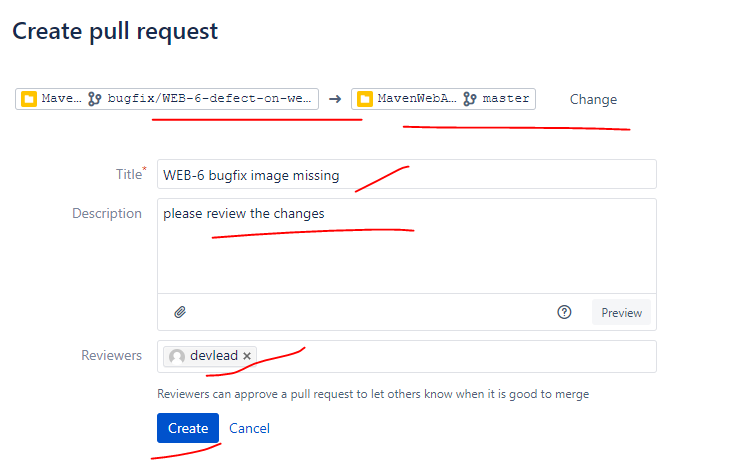


Click on branch to create pull request:

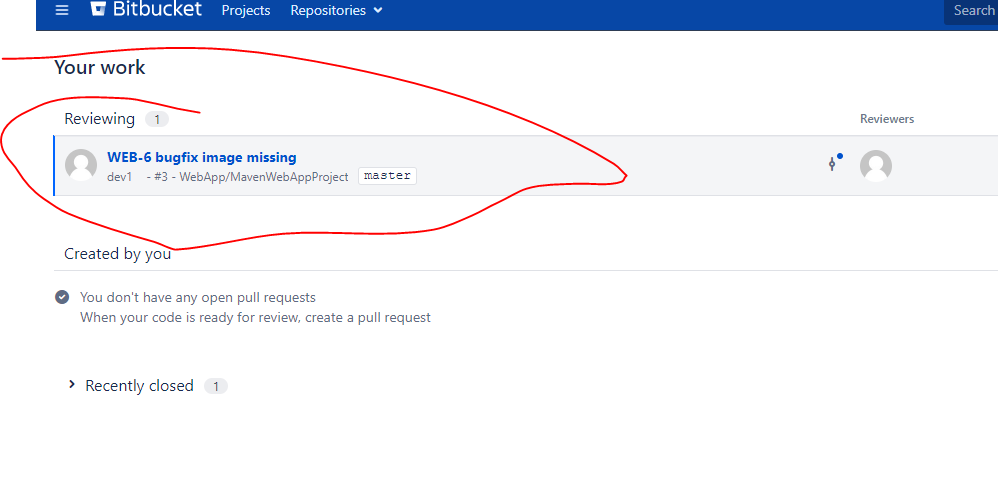




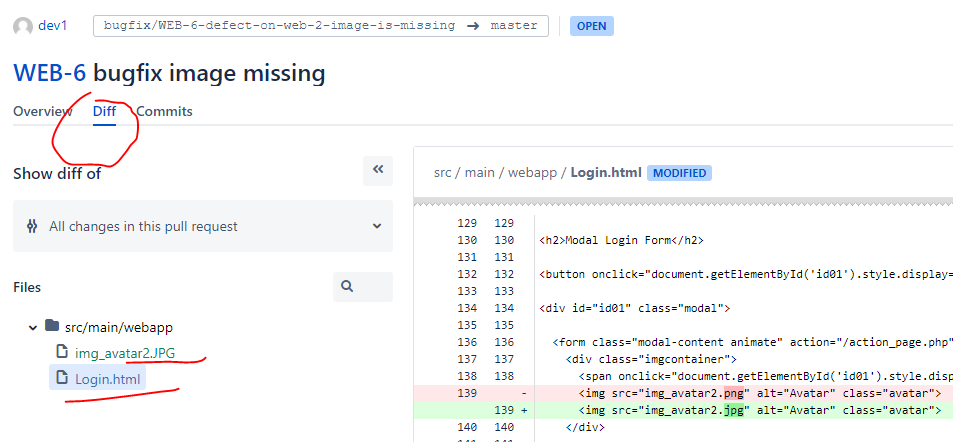
Assign the PR to review by devlead:



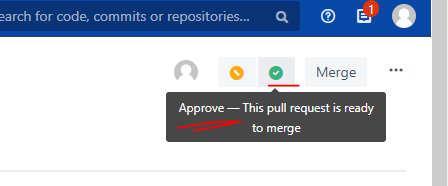
Pending reviews for devlead user:



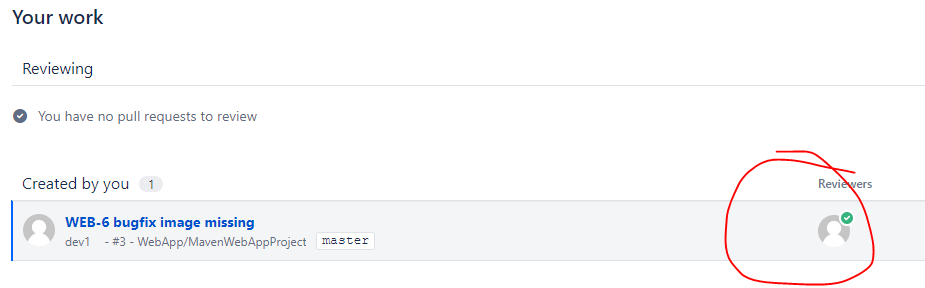
Devlead is Reviewing the changes on the files:



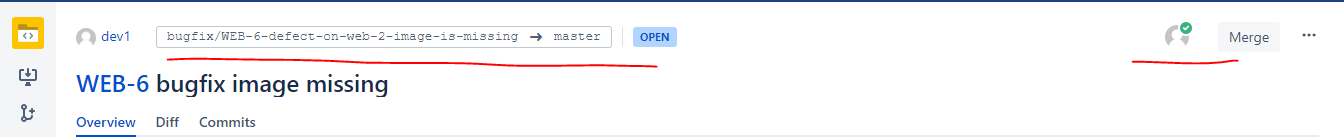
Approving the PR after review:

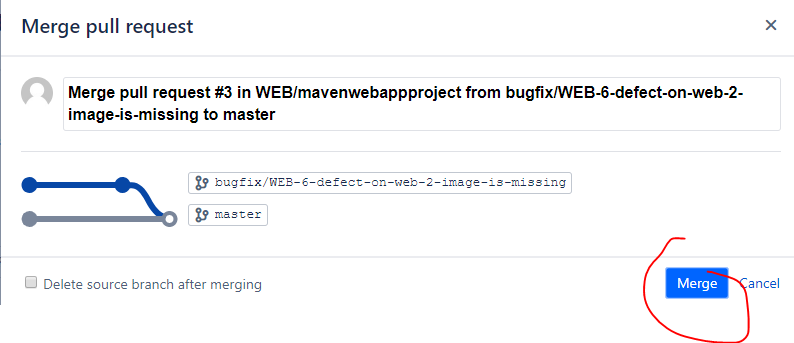


Approved PRs list for **dev1** user:

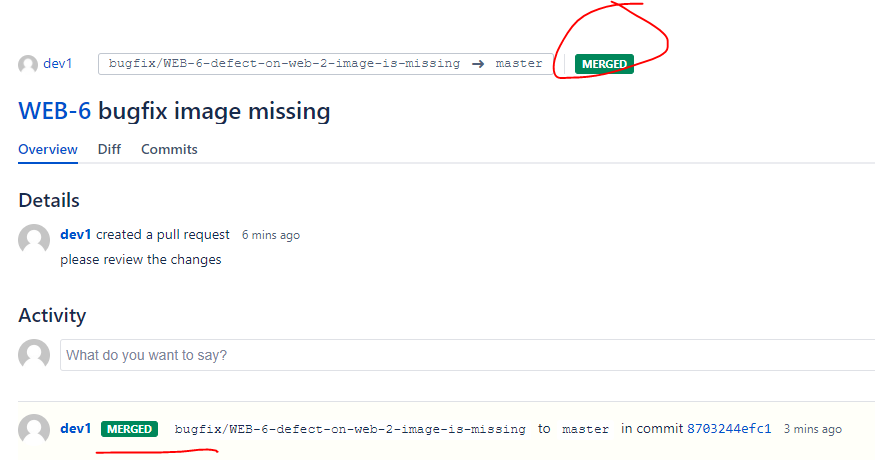


Proceed with merge as it is approved by devlead:

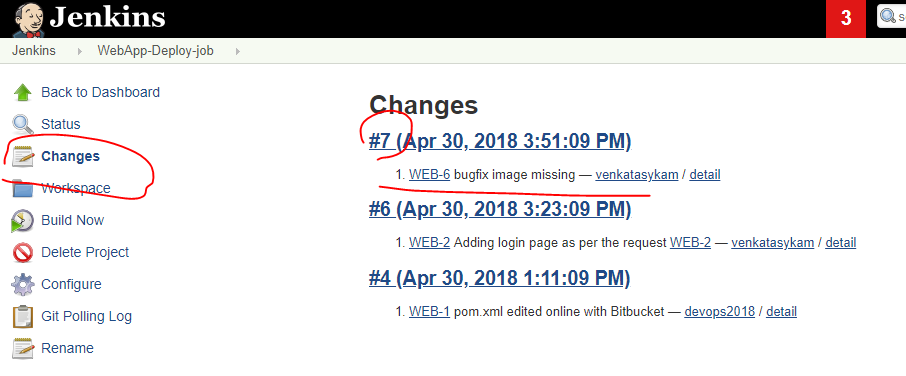


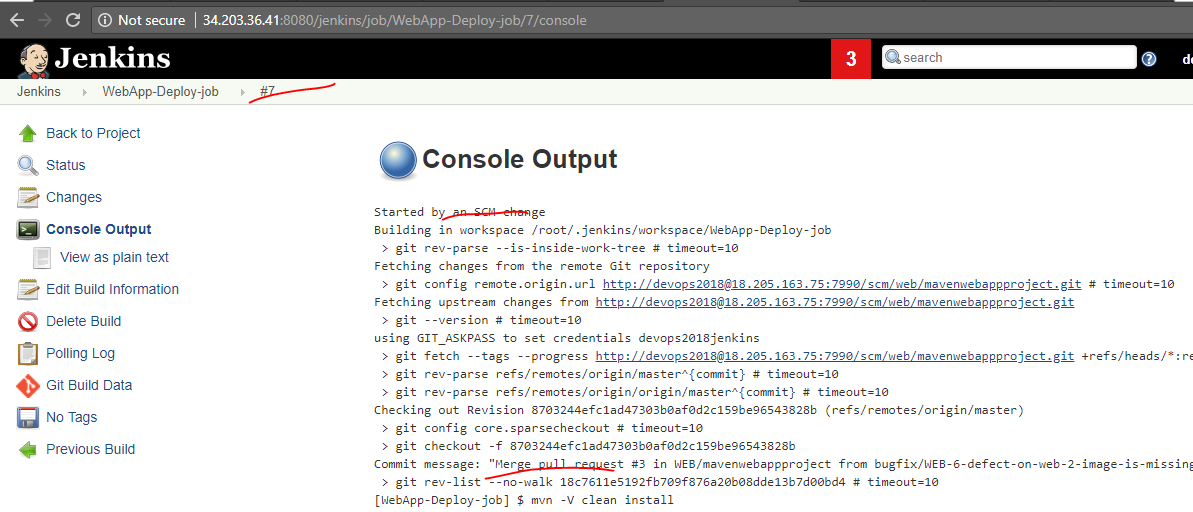


Merge success:



Go to Jenkins and check the automated build:





Build and deployment success. If we are getting expected report then proceed with new development. If there is any issue, open a bug and update the existing code to fix the issue. In this example, we are continuously adding code that means the code is integrating by all the changes continuously i.e., Continuous Integration.

As shown in the below image, we can implement CD with actual test environments.

