****

**Git Check-in Process & Lifecycle**

Created by: eTouch Systems Corp.

Dt. 4/26/2016

eTouch Systems Corp.,

6627 Dumbarton Circle, Fremont, CA 94555

Tel: 510-795-4800x 117 Fax: 510-795-4803

**Revision History:**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description of Change** |
| V1.0 | 4/26/2016 | Initial version |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[**1.** **GIT CODE MERGE PROCESS** 4](#_Toc449453259)

[**2.** **GIT CHECK-IN LIFECYCLE** 6](#_Toc449453260)

# **GIT CODE MERGE PROCESS**

1. Developer clone’s remote development repository **test\_automation\_library** (**origin**) to local. *(Required only for first time setup)*



1. After the completion of cloning all remote branches should be available in local with default selected branch as **master**.
2. A developer gets assigned to a task with some end goals for example the task is “**Fix Firefox issues**”.
3. To start the feature development, developer creates a new branch **firefoxFixes** using **master** as base.



1. Developer completes the changes and commit’s the changes as and when is required.



1. Switch to **master** and get the latest **master** from origin.





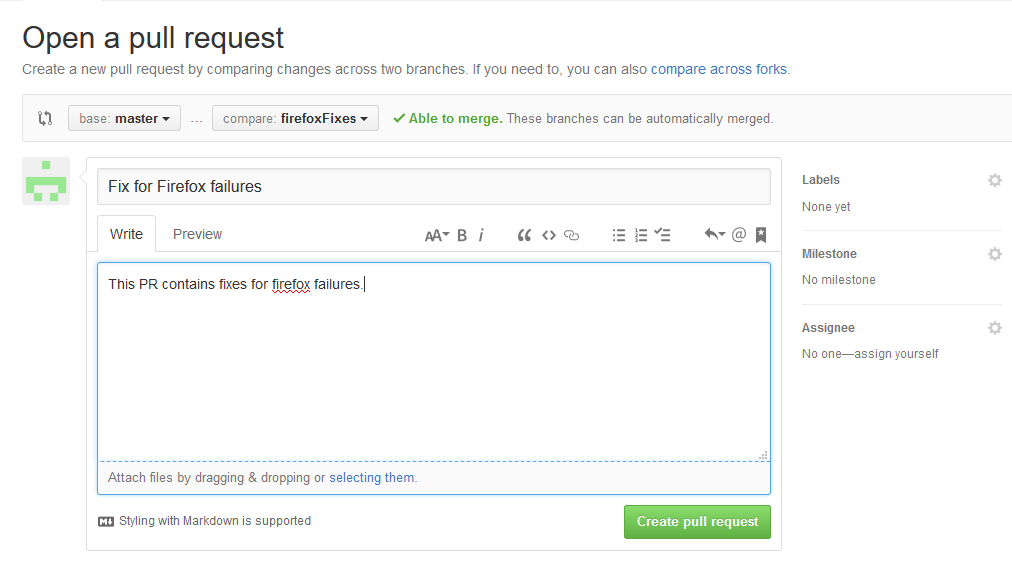
1. Switch to **firefoxFixes** branch, merge **master** and resolve conflicts.



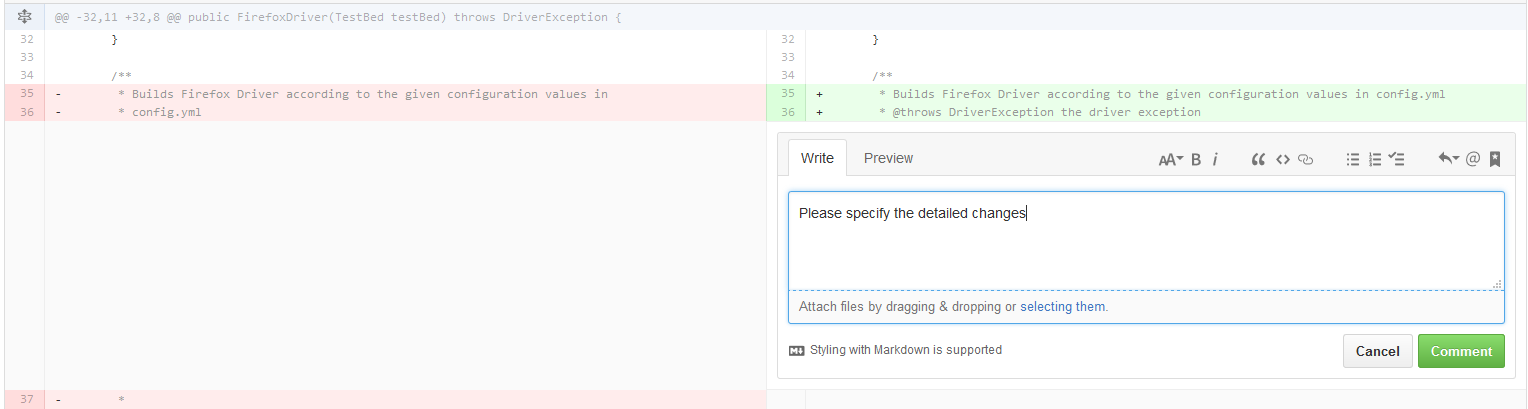
1. Run the unit test cases to check if any new issues are introduced.
2. Push the **firefoxFixes** branch to origin.



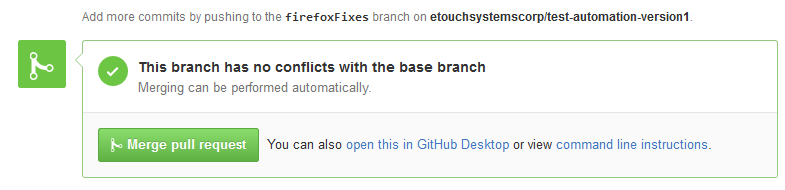
1. Raise a code pull request (**PR**) against **master**/respected branch.



1. Reviewer reviews the **PR** and gives the comments and also checks if Jenkins run is required.

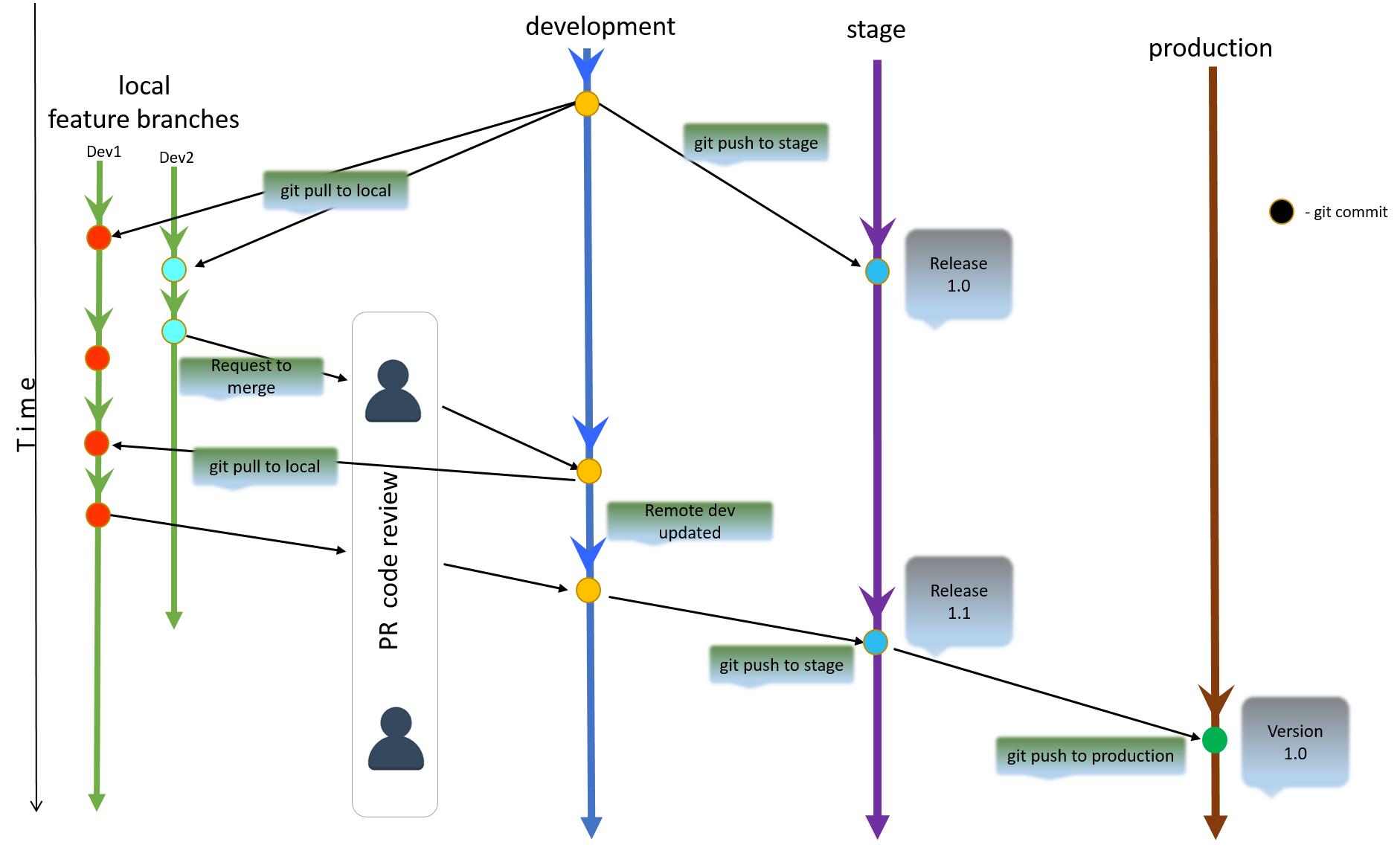


1. Developer incorporates the request changes, commit’s them and pushes to remote **firefoxFixes** branch.
2. Runs the Jenkins job as per the comments at 11.
3. **PR** gets merged in Master once clean is achieved and review comments are incorporated.



# **GIT CHECK-IN LIFECYCLE**

Below is the flow diagram which shows how code gets merged into master and then gets deployed to Stage and Production environment.



**Detailed flow:**

1. Dev1 7 Dev2 starts working on 2 different features by pulling the latest code to their local branches from remote Development repository.
2. Dev 1 makes 1 commit in his code and continue working on the feature.
3. In the meantime, Dev2 makes 2 commits and finishes the development and raises a Pull Request (PR) for the merge of completed feature.
4. Repository admin gets notified about the new Dev2 -PR and reviews the code.
5. Admin merges the code in master as they find that the code complies with all the requirements.
6. Code in Developments repository gets updated with the latest changes.
7. Dev1 finishes the development of feature and makes a commit in local.
8. Dev1 pulls the latest code to local and merges with the feature branch to make sure that there are no merge conflicts.
9. Once the conflicts are resolved he pushes his branch to remote and raises a PR.
10. Repository admin gets notified about the new Dev2 -PR and reviews the code.
11. Admin merges the code in master as they find that the code complies with all the requirements.
12. As with Dev1 and Dev2 features are completed, Code from Development repository gets deployed to Stage on periodic basis.
13. Once the test on Stage are successful this code deployed to Production as per the schedule.