

**GIT Branching Strategy with Agile Practice**

**Version 1.0**

**© Xavient Information Systems**

This document contains confidential and proprietary information of Xavient Information Systems, the disclosure of which would provide a competitive advantage to others. As a result, no part of this document should be disclosed, used, duplicated, reproduced, stored, copied, or transmitted, in whole or in part, in any form or means, electronic, mechanical, photocopying or otherwise without the express consent of Xavient Information Systems. This document shall remain the property of Xavient Information Systems.

This restriction does not limit the rights of the recipient to use information contained within the document if it is rightfully obtained from the source without restriction. The data subject to this restriction are contained in the entire document

**Xavient Information Systems**

2125 C Madera Road; Simi Valley, C.A 93065.

Main Line: 1.805.955.4111, Fax Line: 1.805.955.4144 Visit us: www.xavient.com

**Version Summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version No.** | **Prepared By** | **Reviewed By** | **Approved By** | **Release Date** |
| 1.0 | Neeraj Virmani | Neeraj Virmani | Neeraj Virmani | 30/06/2017 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Modification Details**

|  |  |  |
| --- | --- | --- |
| **Version No.** | **Release Date** | **Description Of Change (if any)** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[1 Why Branching and Merging strategy 4](#_Toc468388396)

[2 Agile based Git Branching and Merging Strategy 5](#_Toc468388398)

**Why Branching and Merging strategy:**

**Branching’s** main purpose is to facilitate parallel development. Each feature will be developed in separate branch without affecting each other. Another frequent reason is functional – branches are created for new features, bug fixes, spikes, releases etc.

* Once a feature is ready for production, you need to take the changes in the branch and apply them to other branches.
* Toggle feature provides an alternative to enable, disable or pick feature that is not ready for production and those can be carry forward to next sprint.
* CI & CT happening in parallel on each and every feature branch helping agile workflow and robust software.
* Tests run against integrated codebase.
* Developers pick each other’s changes immediately.
* Modern version control tools make merging non conflicting changes relatively easy to merge. The problem arises when conflicting changes has been made in branches that you are trying to merge. **That’s why we need a well-defined Branching and Merging Strategy.**

IT industry is adopting Agile software development methodologies and there is no well-defined Branching and Merging strategy is available which compliments Agile.

**Agile based Git Branching and Merging Strategy**

Below blueprint demonstrate the process of working with Agile’s scrum process having a sprint of 15 days each.

1. INITIAL DAY ( SPRINT 1 ): Day 1

* First create a **Development** Branch from **Master** Branch**.**
* Create a separate Feature branch for individual features (i.e. **F1, F2…Fn**) from **Development** branch and start working on functionalities.

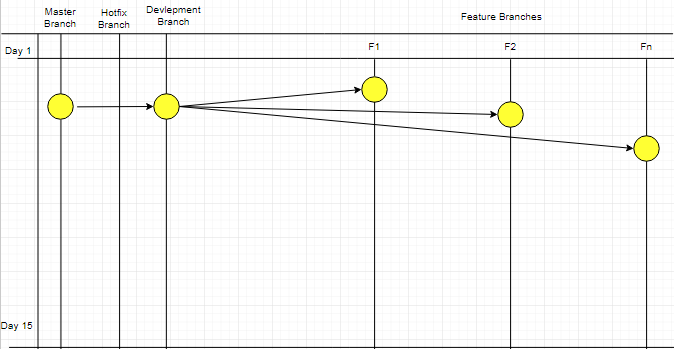


Figure 1

1. INTERMEDIATE DAYS( SPRINT 1) : DAY 2 - DAY 13

* Targeted work is complete in feature branches **F1** and **F2** and the changes are “Ready to Go”.
* **Fn** is still being worked on and not ready to be shipped at this stage.

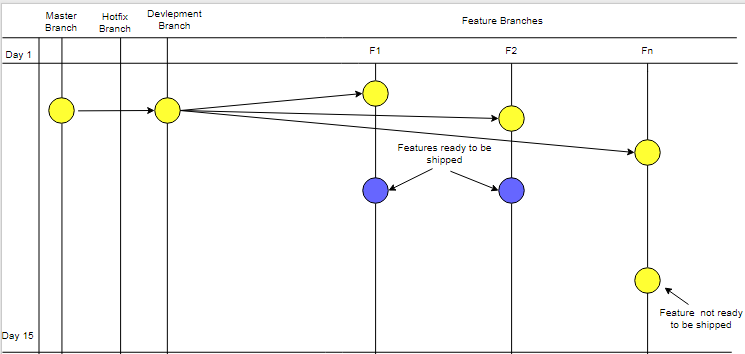


Figure 2

* Create a branch namely **F1F2**
* Merge branch **F1** into **F1F2.**
* **Fn** is still not ready to be shipped and being worked on.

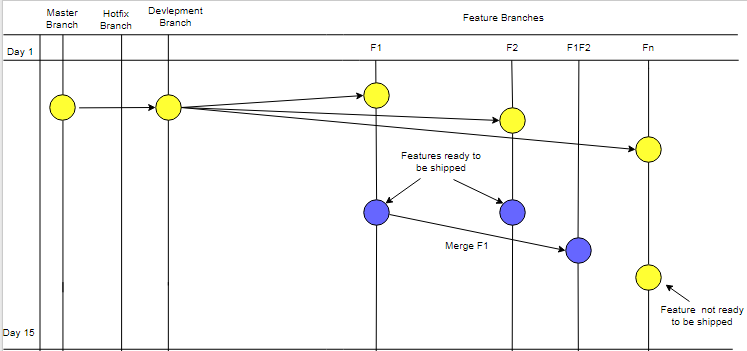


Figure 3

* Similarly merge branch **F2** into **F1F2.**
* **Fn** is still not ready to be shipped and being worked on.

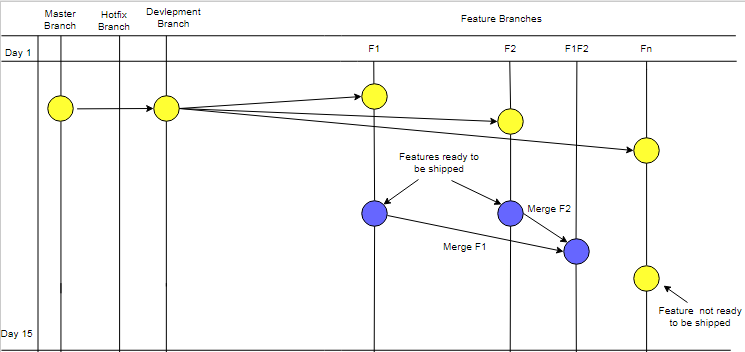


Figure 4

* Merge **F1F2** back into **Development** branch.
* **Fn** is still not ready to be shipped and being worked on.

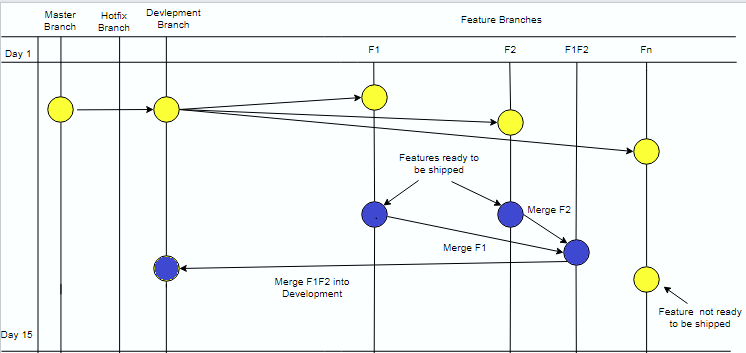


Figure 5

* **F1F2** is ready for production within current sprint.
* Merged **F1F2** into **Development** branch.
* **Development** branch will be merged back to the **Master** branch with a **tag v1.0.0**.
* **Fn** is still not ready to be shipped and being worked on.

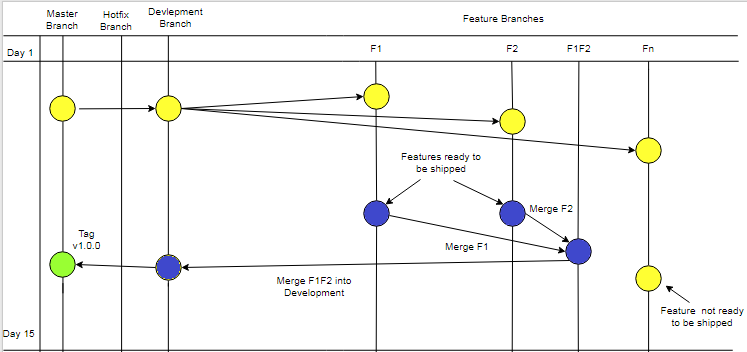


Figure 6

1. FINAL DAYS (SPRINT 1) :DAY 14-15

* Create a **tag v1.0.0** from **Development** branch and merged in **Master** branch.
* Delete branches **F1**, **F2** and **F1F2** as we have merged “ready to be shipped” features into **Development** branch.
* **Fn** is still not ready to be shipped and being worked on.

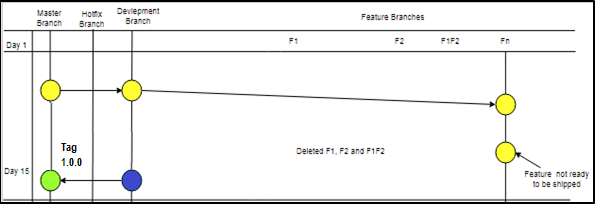


Figure 7

* If we encounter any bug in Production we create a **HotFix** branch from **tag v1.0.0.** to make that small change and start working on the bug.
* **Fn** is still not ready to be shipped and being worked on.

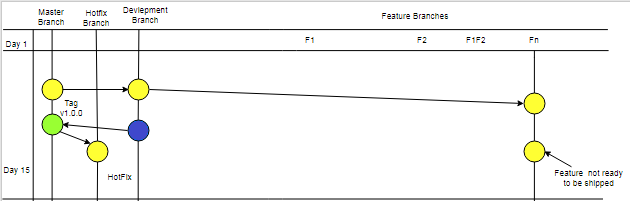


Figure 8

* Fix the encountered bug in **Hotfix** branch.
* Create the new tag **v1.0.1** and merge back into **Development** branch and **Master** branch simultaneously.
* **Fn** is still not ready to be shipped and being worked on.
* Here our first sprint has ended.

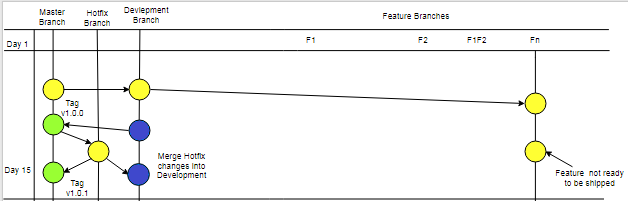


Figure 9

* Feature **Fn** will now be continued into next sprint (i.e. Day 16 onward)

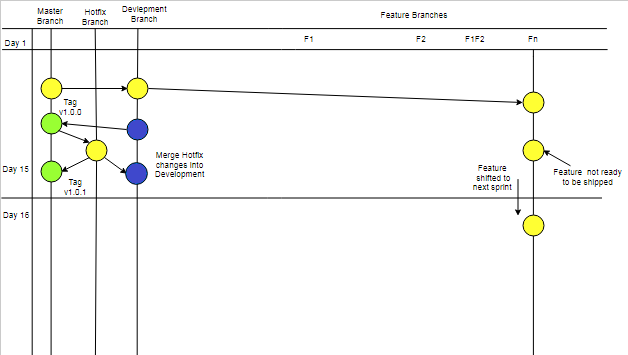


Figure 10

1. INTIAL DAY (SPRINT 2) : DAY 1

* Now feature **F3 (Fn)** gets code merge from **Development** branch (i.e. Day 16).

This strategy will be followed in each sprint.

* New features can also be created for this sprint cycle.

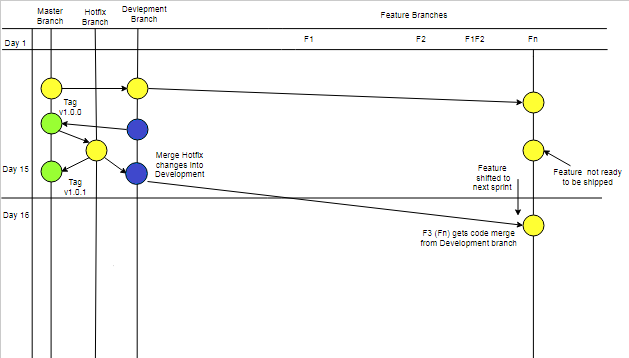


Figure 11