Enterprise Integration Platform

***GitHub branching strategy and protection rules***

Table of Contents

[**Branching strategy:** 2](#_Toc72473092)

[**Main branch** 2](#_Toc72473093)

[**Release branch** 3](#_Toc72473094)

[**Develop branch** 3](#_Toc72473095)

[**Feature branch** 3](#_Toc72473096)

[**BugFix branch** 4](#_Toc72473097)

[**Hotfix branch (Production Fix)** 4](#_Toc72473098)

[**Branch Protection Rules:** 5](#_Toc72473099)

[**Teams/Roles:** 6](#_Toc72473100)

## **Branching strategy:**

There are many different branching strategies available. We will look at one such strategy which will immensely aid the release management.

Initially, the default branch would-be master. We will create the staging/release and develop branches and we will make develop branch as the default branch.

* develop **(default)**
* release
* Main **(production)**

The above branches should be created before developing any features and should not be deleted at any point in time, if the project/application exists.

Some of the rules which are generally followed for the branches.

**Main branch**

* This is the main branch, whose code is deployed in the production environment.
* No new branch can be created from main except HotFix branch.
* No commits can be directly made in main branch.
* It should accept pull requests only from the release branch and it should not accept pull requests from other branches including the develop/hotfix branch.
* Should not have unreleased code [ every commit in master must have a stable release associated]
* The pull request from the release branch should be reviewed and approved by the client/stakeholder/productOwner and must be executed by EIS Operation team member.

**Release branch**

* The code in this branch will be deployed in release (UAT) environment
* No direct commits are allowed.
* It should accept pull request only from the develop branch and hotfix branches.
* It can make a pull request only to the main branch.
* Should not have unreleased code [ every commit in release must have a beta release associated]
* The pull request from the develop branch should be reviewed and approved by the tech lead/architect.

**Develop branch**

* The code in this branch will be deployed in development environment.
* No direct commits are allowed.
* It should accept pull requests only from the feature or bugfix branches and should not accept pull requests from user branches under any circumstance.
* It can make a pull request only to the release branch.
* Should not have unreleased code.
* It is the base branch for feature.

**Feature branch**

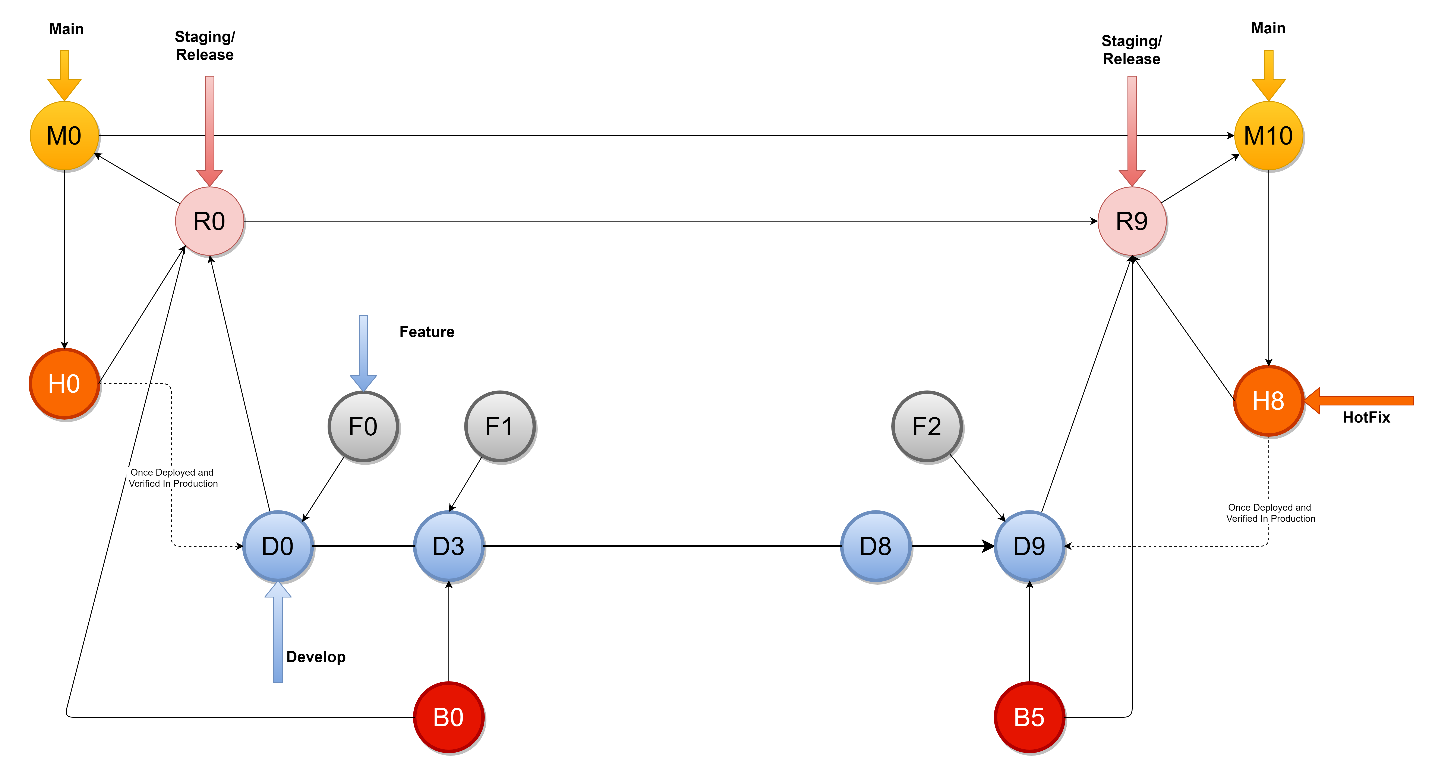
* Any new feature should have its own branch created from develop branch.
* The naming convention to follow for the feature branch is feature/$(x) where x denotes the name of the feature/JIRA story number to be developed.
* Direct commits are allowed.
* It can have unreleased code.

**BugFix branch**

* Any new bugfix should have its own branch created from develop branch.
* The naming convention to follow for the feature branch is bugfix/$(y) where y denotes the name of the bug to be fixed.
* Direct commits are allowed.
* It can have unreleased code.
* We can delete the branch after fixing the bug.

**Hotfix branch (Production Fix)**

* Any hotfix/production fix should have its own branch created from master branch.
* The naming convention to follow for the feature branch is hotfix/$(y) where y denotes the name of the bug to be fixed.
* Direct commits are allowed.
* Upon working on hotfix branch, it should get merged to release branch and deployed to UAT.
* The pull request from the hotfix branch should be reviewed and approved by the tech EIS lead/architect.
* Hotfix branch is deployed to production and verified successfully in production for the fix.
* PR from hotfix branch to develop branch so that the code fix is never missed for future release.
* We can delete the branch after fixing the bug/merging to develop branch.



**Branch Protection Rules:**

|  |  |
| --- | --- |
| Branch | Permissions |
| **Develop** | Branch name pattern: develop |
| Require pull request reviews before merging |
| Dismiss state pull request approvals when new commits are pushed |
| Require status checks to pass before merging |
| Allow force pushes |
| **release** | Branch name pattern: release\* |
| Require pull request reviews before merging |
| Dismiss state pull request approvals when new commits are pushed |
| Require status checks to pass before merging |
| Allow force pushes |
|  | Allow deletions |
| **master** | Branch name pattern: master |
| Require pull request reviews before merging |
| Dismiss state pull request approvals when new commits are pushed |
| Require status checks to pass before merging |
| Restric who can push to matching branches |
|  | EIS Operation |
| Include administrators |

**Teams/Roles:**

* **EIS Developer:** Mule code development and only allowed to publish the code to development environment. Has complete access to development environment.
* **EIS Architect**: complete access on all branches
  + Not allowed to approve PR to master branch.
  + Allowed to approving code/PR for deployment to SIT and UAT environment.
  + Manual deployment to SIT and UAT.
  + Change/update properties on lower environment (DEV, SIT,UAT)
* **EIS Operation:** 
  + Approve PR for deployment to master.
  + Allowed for manual deployment in production.
  + Allowed changes to platform properties in production.