Code Branch and Commit Strategy

Created by: JP

Branching Strategy

- Use feature branches for all new features and bug fixes
- Merge feature branches into the master branch using pull requests.
- Keep a high quality, up-to-date master branch.

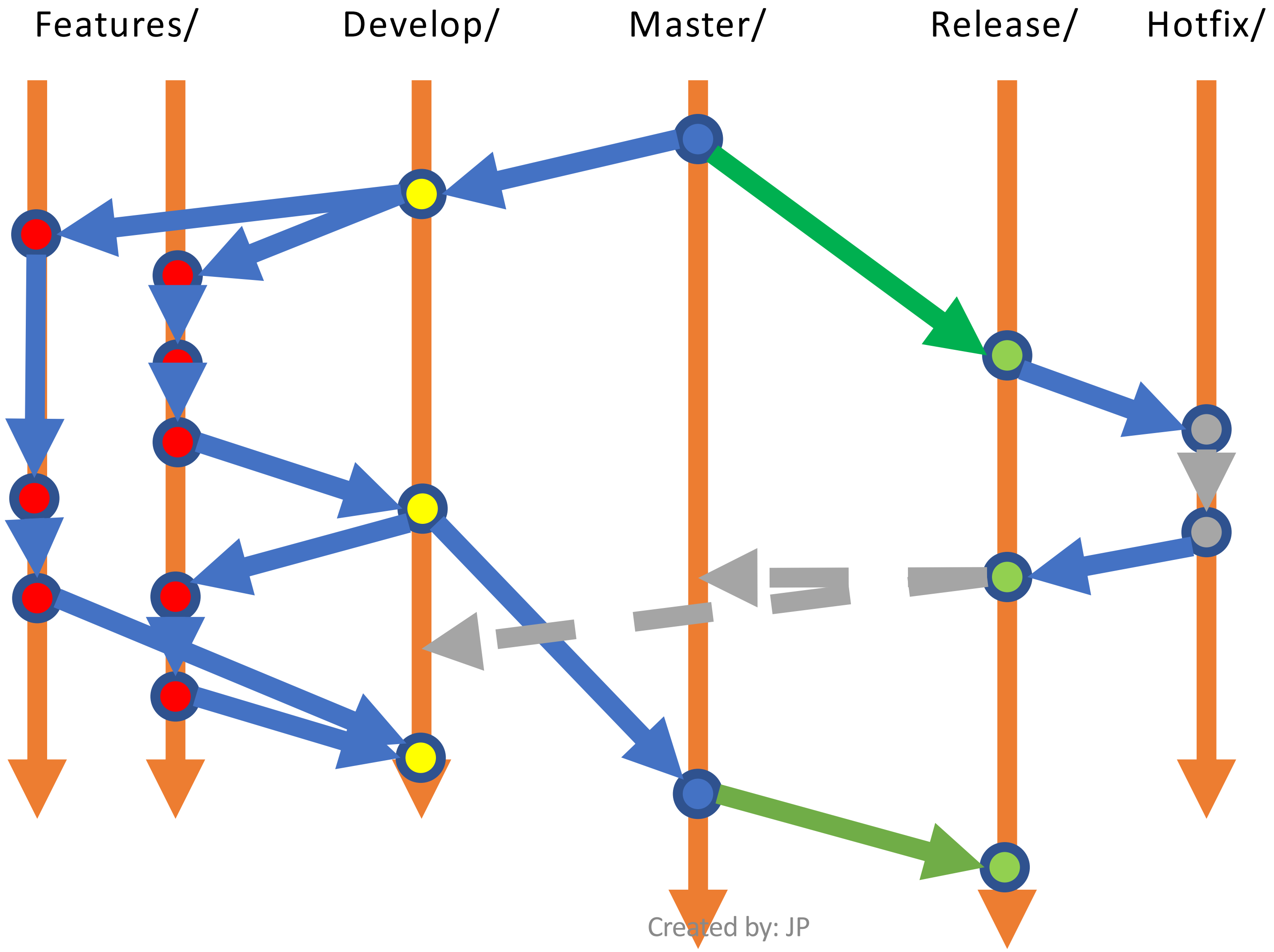
Development Branches

- Ledger would have 3 level of branches
  - Master/
    - This will be the base branch
    - It will be the most recent and up to date branch
    - All live releases will be done from this branch
  - Develop/
    - This branch will host the changes done by developers before merging it to Master Branch
    - Code merged under this branch will be assigned to QA for testing
    - Only upon approval from the QA, a Pull Request will be initiated on this branch
  - Feature/ or User/
    - This branch(es) will be created for development of features
    - In case a user/ developer wants a separate branch for himself, he can create the branch
    - The feature team will be responsible for this branch
    - Upon unit testing and confirmation from the developer, a pull request will be initiated on this branch

Branch Folders

- Each branch created should have a separate directory
- This helps to properly classify/ group the branches

Development Branch Layouts



Created by: JP

Branch Naming Conventions

- Each branch should have a logical and explanatory name
- Feature Branches
  - features/feature-name
- Bug Fixing Branches
  - bugfix/description
- Develop Branches
  - develop/sprint-name

Pull Request and Code Review

- Pull requests to be generated by developers while committing the code to the parent branch
- Each pull request to have a Reviewer assigned
- Code to be reviewed by the reviewer
  - Approve
  - Reject
- Approve: Code to be merged into the parent branch
- Reject: Code will not be merged and developer will have to fix the issues/ concerns

Pull Request

- Pull request should be reviewed on the Web Portal of TFS
- Comments and merging should be done from the web portal itself

Gated Check Ins

- Gated check ins will be implemented at
  - Develop/ → Master/ branch Commits
  - Master/ → Release/ branch Commits
  - Hotfix/ → Release/ branch Commits
  - Hotfix/ → Develop/ branch Commits

Release Branching

- Releases will never be done on Master/ branch
- A separate branch will be pulled from Master/ for release
- Auto Build and Release process will be implemented on this branch
- The release will be triggered manually

- In case there are issues identified in the release, a Hotfix/ branch will be created and issues fixed there
- The fixes will then be ported to Master/ and Devlop/ branches

Release Branch Naming Convention

- Each branch should have a logical and explanatory name
- Release Branches
  - release/release-name
  - release/release-number
- Hotfix Branches
  - hotfix/release-name

Auto Build Implementations

Source Branch	Destination Branch	Check In Type	Check In Review	Build Type	Release To
Local	Feature/ or User/	Manual	No	Manual	QA Environment
Feature/ or User/	Develop/	Manual	Yes (Using Pull Request)	Auto Build	QA Environment
Develop/	Master/	Manual	Yes (Using Pull Request)	Auto Build	Pre Live Environment
Master/	Release/	Manual	Yes (Using Pull Request)	Auto Build	Live Environment
Hotfix/	Release/	Manual	Yes (Using Pull Request)	Auto Build	Pre Live Environment

Reference Reads…

- Git Naming Standards
- VSTS Git Branching Guidance
- Git in VSTS
- VSTS Git Tutorials (Videos)