**ITWorks Version Control System practices for Git/GitHub:**

* Name the repository. Use Pascal case convention. e.g. SalesAppRepos
* Name branches in lower case and underscore between words. Common branches are: master, development, feature\_xxxxx, release, bugfix\_xxxxx.
* For git local repository, the default branch is the “**master”** branch. For ITWorks, for consistency between projects, name the default branch is named “**master**”.
* The master branch is a permanent branch which always reflects a production-ready state. Create tag to identify the version such as v1.0.0, v1.1.0.
* Git repositories for software development should have README.md and LICENCE.md file.
* Git repositories should have .gitignore file at the root directory of the project. The .gitignore file specifies intentionally untracked files that Git should ignore. (i.e. with appropriate filters for C# program)
* After the installation of the Git for Windows in the local workstation, the git server must be configured with a user name and email address.
* Commit files that belong to a task in a single purpose to keep the project consistent at all times.
* Each commit should have included a descriptive commit message which reflects / summarizes the codes being changed.
* Review the codes before committing to the repository.
* Share your changes frequently. i.e. after commit locally, pull first and push the changes to the remote repository to share with other team members. It's important to run the git pull command before you push any new changes to the remote branch. This will update your local branch with any new changes that may have been pushed to the remote from other contributors. Pulling before you push can reduce the amount of merge conflicts you create on GitHub - allowing you to resolve them locally before pushing your changes to the remote branch.
* A commit is often associated with a specific issue or new product feature. The project should be able to build successfully before and after the commit.
* All development works (bugfix or new features) must be carried on the branches. Each branch should have defined who can make changes, when builds occur, and when test to be run etc.
* Branches should be short-lived. Remove them when they are no longer needed. This reduces the risk of using or merging to the wrong branch.
* Release branches are created from the development, bugfix\_xxxx, feature\_xxxx branch.
* Avoid indiscriminate commits – meaning commit without a specific file. TotoriseGit do not allow to do this anyway.
* For GitHub remote repository, the default branch should be called “**main**”. Recently the GitHub changed the default branch from “**origin**” to “**main**”. You can change the default branch name at the time you create the remote repository.
* When using the client tools to interface with the remote server, industry communication protocol must be used (e.g. RSA, SSH. etc for private/public key pairs) to protect the security of in the communication between the client and the remote server.

**ITWorks Git workflows practice in general:**

master release development feature1 feature2 ……..

Add README.md, LICENCE.md, .gitignore files

Add some basic codes before major feature develooped through branches.

Initial loading of codes

Develop codes in feat 1 branch

Develop codes in feat 2 branch

Merge to release branch

Merge to master branch and create tag Vxx.x.x

The above workflows only shows a general version control project with the basic workflows using some common branches.

* **master branch** - default branch. It is always be the latest stable version of the project to be deployed to the production environment
* **development branch** - created from master branch. The development branch gets the code from the master branch for development in a sprint. The feature branches merge into the development branch.
* **feature branch(es)** - created from development branch. They are used for developing new functions/features.
* **release branch** - created from the development branch. It is a sprint specific which is submittebale to the master for production. It is used for the final testing before merged to the master branch.
* **bigfix branch** – use for just for a small set of changes to be released or build which is required to fix a bug. This branch is optional.

The above workflows is only actr as a general guides. Individual project may vary with more or less the branches. However, the master, development, release and feature branches are most likely be used.