GIT (Version Control)

Introduction :

Git is a distributed version control system. It allows the users to have versions of a project, which show the changes that were made to the code over time, and allows the user to backtrack if necessary and undo those changes. In the projects where more than one developer works,it uses the idea of a centralised version control server where all versions are stored on a central server, and individual developers checkout and upload changes back to this server.
Git specifically works by taking snapshots of files; if files remain unchanged in a particular version, it simply links to the previous files - this keeps everything fast and lean.

Functional Features:

1. Store and view the versions of the project.
2. User finds the Difference between the version.
3. Centralised server to store the projects of all the developers.

Non-Functional Features:

1. Distributed Nature:

In a distributed VCS like Git every user has a complete copy of the repository data stored locally, thereby making access to file history extremely fast, as well as allowing full functionality when disconnected from the network. It also means every user has a complete backup of the repository. So, the probability of loosing data is less.

2. Branch Handling:

Branches are so core in Git because every developer's working directory is itself a branch. If two developers are modifying two different unrelated files at the same time it's easy to view these two different working directories as different branches stemming from the same common base revision of the project.

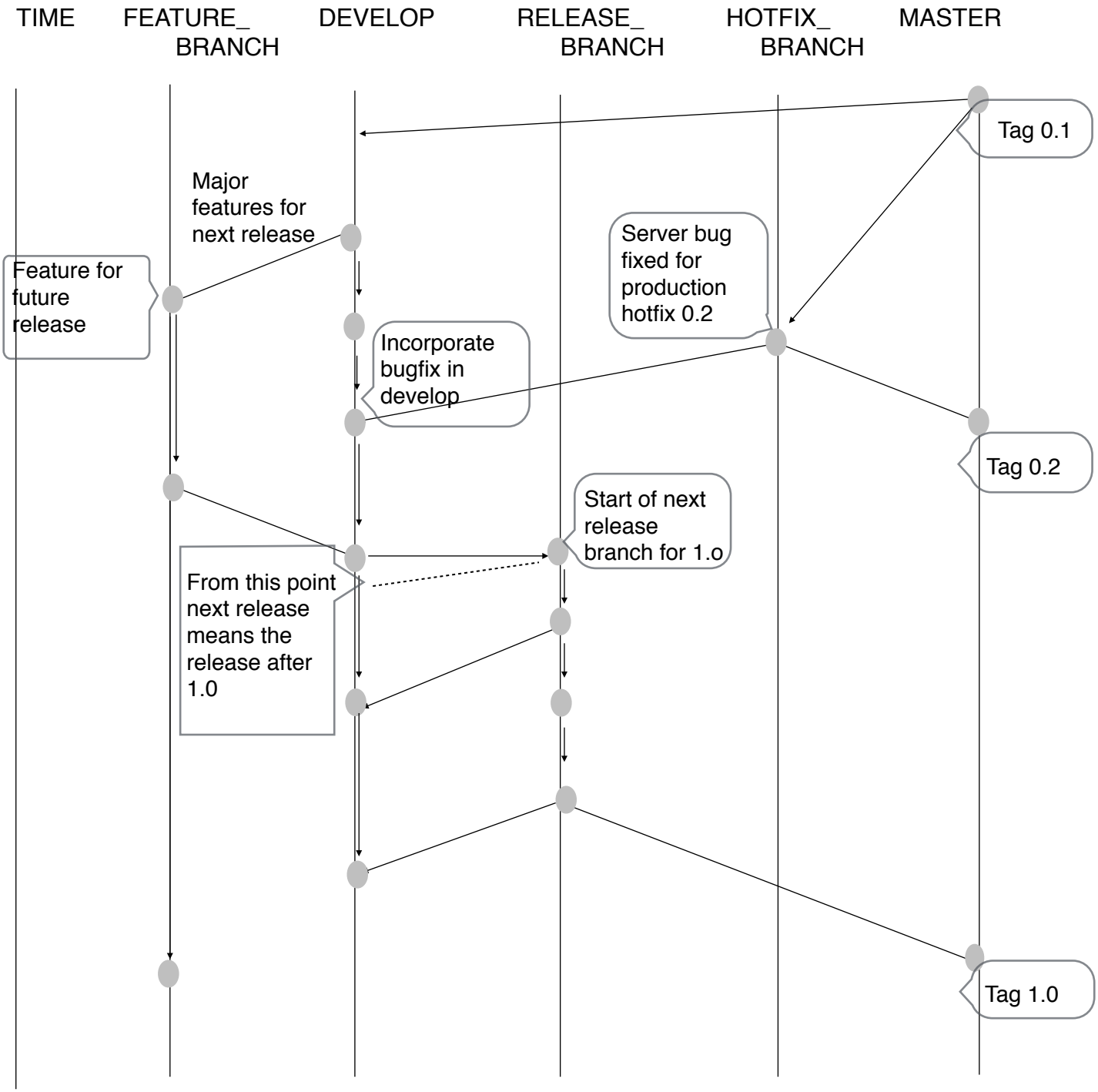
3.Performance:

Git is extremely fast. Since all operations (except for push and fetch) are local there is no network latency involved to view file history, commit changes, merge branches, obtain any other revision of a file and to switch branches.

4.Smaller Space Requirements:

Git's repository and working directory sizes are extremely small. Git working directory requires only one small index file that stores about 100 bytes of data per tracked file. On projects with a large number of files this can be a substantial difference in the disk space required per working copy.

Design Architecture:



1.Main branches:

The central repository holds two main branches namely **Master** and **Develop.**
Master is the main branch where the source code of HEAD always reflects a *production-ready* state.
Develop is the main branch where the source code of HEAD always reflects a state with the latest version.
When the source code in the develop branch reaches a stable point and is ready to be released, all of the changes should be merged back into master.

2. Supporting branches:

Git uses a limited life time supporting branches to aid parallel development between team members, ease tracking of features and to assist in quickly fixing live production problems.

a. Feature branches:

Feature branches are used to develop new features for the upcoming or a distant future release. It doesn't do anything particularly special – it just creates a branch in which you can develop a feature. The essence of a feature branch is that it exists as long as the feature is in development, but will eventually be merged back into develop or discarded. It should contain a small unit of work and that there can be many feature branches being worked on independently

b. Release branches:

The purpose of this branch is to test, stabilise and finally release the changes.
The key moment to branch off a new release branch from develop is when develop reflects the desired state of the new release. At least all features that are targeted for the release-to-be-built must be merged in to develop at this point in time. All features targeted at future releases must wait until the release branch is branched off.

c. Hotfix branches:

Hotfix branches are very much similar to release branches. This branch exists to allow a single defect to be fixed, tested and then merged back into the master branch. This is not a process for adding extra features into the production release – this should only be used to fix a specific defect – any other features should be handled in the normal way using feature branches. The essence is that work of team members can continue, while another person is preparing a quick production fix.