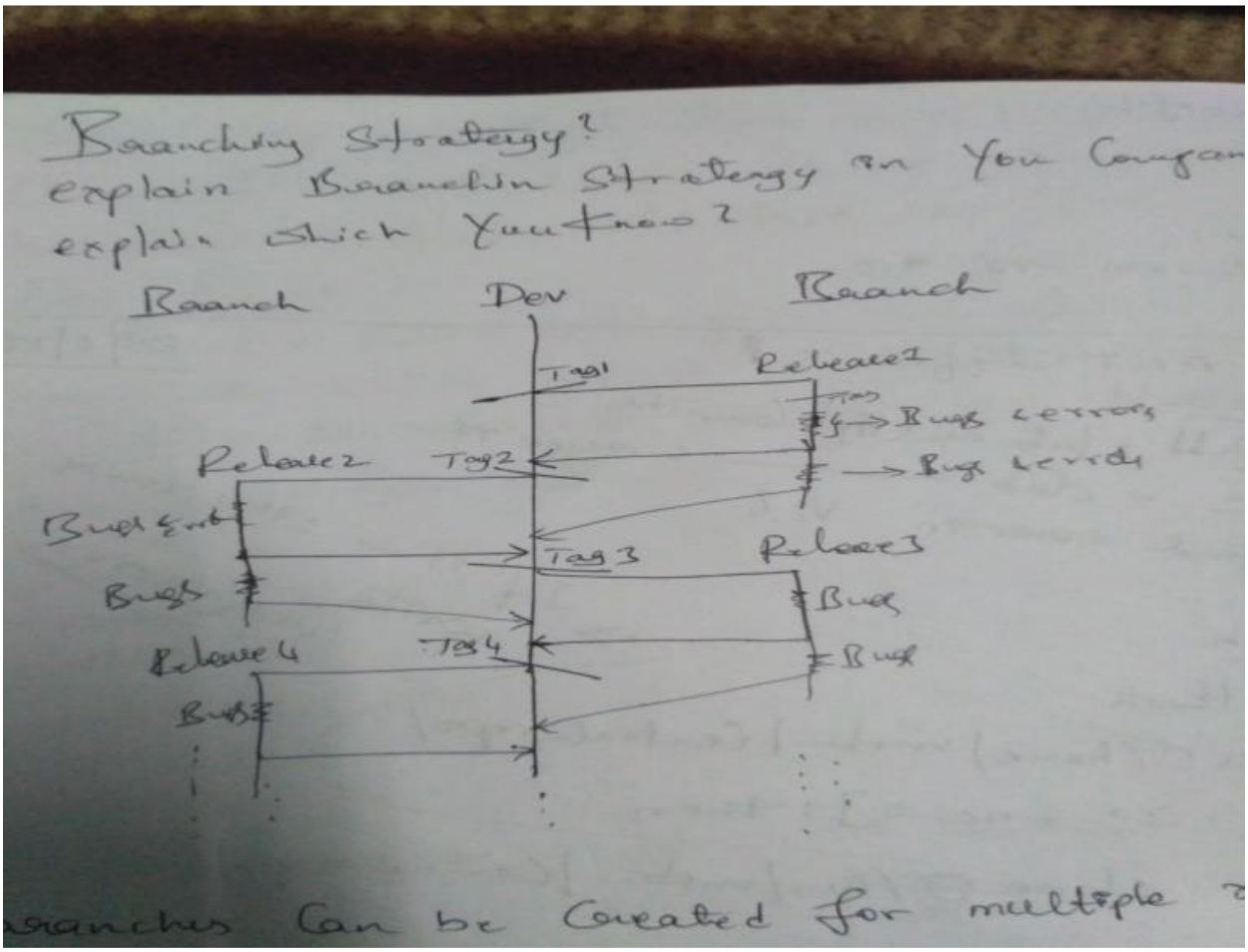
Q1. Branching Strategy or explain branching strategy in your company, explain which u know?

Branching example1



Branches can be created for multiple reasons, here we create branches for releases.

o Development team will be going on development branch, once the code is ready for the first release, we create a release1 branch from dev branch and we make a release (we do build and release it) from this release1 branch.

o Whatever the issues specific to this release will be foxed on release1 branch only. It will act as a maintenance branch for release1.

o Simultaneously development will be going on dev branch for 2nd release. Once the code is ready for 2nd release before we create a branch for 2nd release, we merge, release 1 branch to dev branch and then we create release2 branch for dev

branch for 2nd release. So what ever the issues we have seen in 1sr release should not be visible in the 2nd release and so on.

- Build→build success→BVT or sanity test→sanity report→release note→test team
- Build→build fail→dev team

o BVT(build verification test) or sanity test.

o Is a basic functionality of a build which should never break.

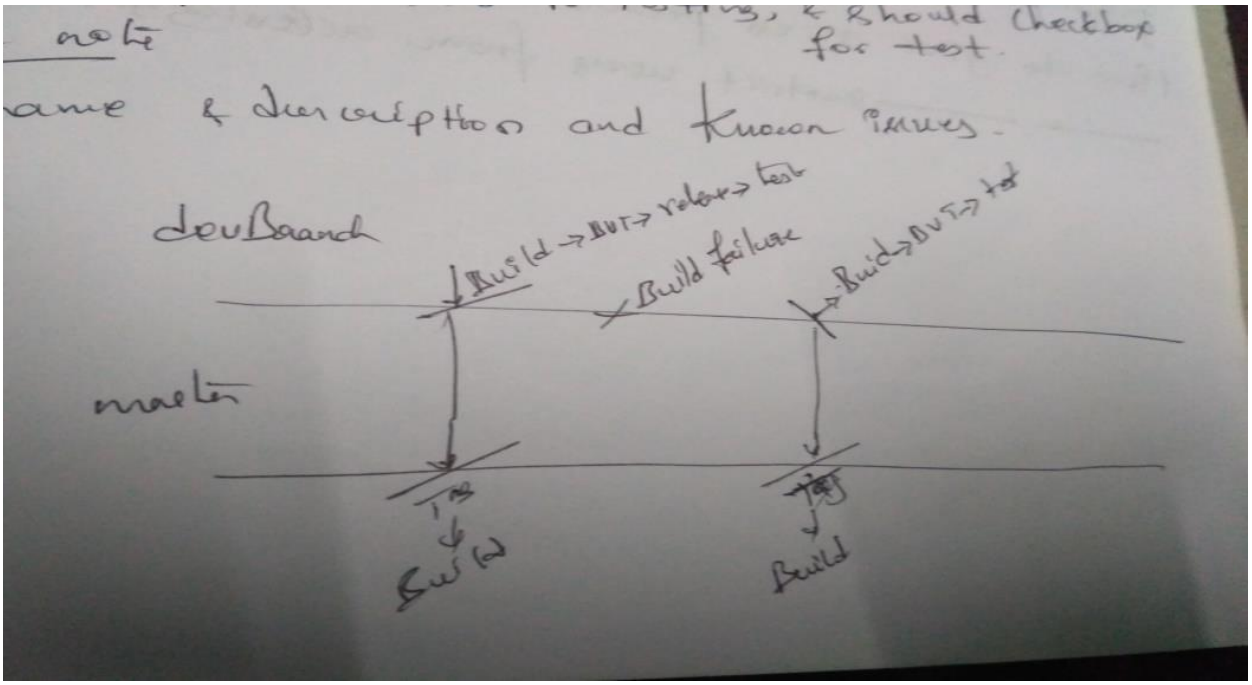
o Sanity report.

- Is a report related to testing and should checkbox for test.

o Release note

- Tag name and description and known issues.

o Branch strategy example2



We have 2branches, one is dev branch another one is master or production branch, developers are allowed to commit changes on dev branch. Once developers

commits the code on dev branch. We build it, we do sanity or BVT and we release to a testing team. We merge this code to production branch. If the build is failed after developer commits the changes, we don‘t merge the code from dev branch to production branch. We work with development team to fix the issue. So we always merge good code to productive branch so that production branch will have clean working branch always. If developer needs latest good code they can pull it from production branch, but they cant push it to production branch, because git push is restricted.

Q2. Describe the branching strategies you have used.

- **Feature branching –** A feature branch model keeps all of the changes for a particular feature inside of a branch. When the feature is fully tested and validated by automated tests, the branch is then merged into master.

- **Task branching –** In this model, each task is implemented on its own branch with the task key included in the branch name. It is easy to see which code implements which task, just look for the task key in the branch name.

- **Release branching –** Once the develop branch has acquired enough features for a release, you can clone that branch to form a Release branch. Creating this branch starts the next release cycle, so no new features can be added after this point, only bug fixes, documentation generation, and other release-oriented tasks should go in this branch. Once it is ready to ship, the release gets merged into master and tagged with a version number. In addition, it should be merged back into the develop branch, which may have progressed since the release was initiated.