**SourceTree:**

1. GIT Client Tool to create branches as per the story/feature in JIRA.

**Jenkins:**

1. Jobs are created to deploy the to QA and Production Environment which are triggered manually.

* **Whenever a release was done the developer has to wait for commit/push on the develop branch until the release is completed as there were conflicts, so how can tackle such situation to continue the deployment without making the developer to wait.**
  + On finishing the Release (FINISH Release on SourceTree), it will merge the changes to the master branch and develop branch, which may have merge conflicts (which is expected) on develop branch, those conflicts need to be resolved manually only.
* **What is the best practice for deploying an artifact to Production Environment (either from release branch or master branch)**
  + Ideally, the deployment should be done from master branch to production environment, as on finishing release (FINISH Release in the source tree) the commit is tagged and merged into master and development branch and the release branch is deleted. So next time to refer any release tag we need to check on the master branch so as the best practice we should be doing the deployment to production environment from master branch.
* **How to accommodate selected new features into release before being deployed to Production environment?**
  + This can be achieved by **Git Cherrypick**

Cherry picking in git means to choose a commit from one branch and apply it onto another.

Make sure you are on the branch you want to apply the commit to.

**git checkout master**

Execute the following:

**git cherry-pick <commit-hash>**

Note:

If you cherry-pick from a public branch, you should consider using

**git cherry-pick -x <commit-hash>**

This will generate a standardized commit message. This way we can still keep track of the origin of the commit and may avoid merge conflicts in the future. For more details please refer [here](http://think-like-a-git.net/sections/rebase-from-the-ground-up/cherry-picking-explained.html).

* **What is the best practice and solution to rollback/revert incorrect release which has already been deployed to Production environment?**
  + This can be achieved by **Git Revert**

A revert operation will take the specified commit, inverse the changes from that commit, and create a new "revert commit". The ref pointers are then updated to point at the new revert commit making it the tip of the branch. This prevents Git from losing history, which is important for the integrity of revision history and for reliable collaboration.

**git revert <commit-id>**