Git Automation Tutorial

Create GIT environment:

1. Create git automation folder **C:\GIT_AUTOMATION.**
2. **cd C:\GIT_AUTOMATION**
3. **git clone ssh://git@bitbucket1:7999/automation/dart.git --branch main**
4. **cd dart/** # now you can see you are on local branch main and your remote branch is origin/main.
5. Add all the files from gitignore to the environment.

Git workflow with commands.

1. Create a new local branch for any feature/bug task.
   git switch -c bugX
2. Fix the bugX in your local directory.
3. When finished to work on the bug – stage all the files you made changes.
   git status
   git add <changed files >
   git status (**changed files** should be in a staged mode).
4. Commit the changes.
   git commit -m 'AUTOMATION-<NUMBER> fix bug X'
   git log -n 3 (validate your commit created).
5. Switch to main branch.
   git switch main
6. Validate you up to date with remote repo main branch.
   git pull origin main
7. Merge bugX branch with local branch main.
   git merge bugX
8. Fix conflicts if need.
   1. Open each file with a conflict (Pycharm)
   2. Fix the conflict.
   3. Commit all the files with conflicts.
      git add <files with conflicts>
      git commit -m 'AUTOMATION-<NUMBER> fix bug X merge conflicts'
      git log -n 3
9. Update remote repository branch main.
   git push origin main
10. Delete local branch bugX when finished.
    git branch -d bugX

Git commands:

git status

Description: git status shows the branch state and all the changes have been made.

git log

Description: git log shows the branch last commits.
Options:

git log
git log -n <number>
git log --oneline

git reset

Description: git reset deletes changes on a branch.
Options:
git reset --hard HEAD (delete the changes for all files)
git checkout filename (delete the changes for a specific file)
git reset HEAD~<number> (delete the latest n commit in your branch)

git add <files>

Description: git add pass the files from modified/untracked mode to staged mode – this is must in order to commit the changes.

git restore –staged <staged files>

Description: git restore turn back the files from staged mode to modified/untracked mode.

Git commit -m <message>

Description: git commit creates a new commit in your local repo branch, which means that the changes are now store in the local repository branch.

git merge <branch name>

Description: git merge integrates a branch to the current checkout branch, which means that all the changes (commits) from other branch will be added to the current branch.

git stash:

Description: git stash locally stores all the most recent changes in a workspace and resets the state of the workspace to the prior commit state. A user can retrieve all files put into the stash with the git stash pop and git stash apply command.
Options:
git stash save "<message>"
git stash list (show the stack stashes)
To apply the stash use:
git stash pop (last stash)
or
git stash apply stash@{<index>} (specific stash)
git stash drop stash@{<index>} (delete stash)

Git cases:

case:

```
$ git switch bugY
error: Your local changes to the following files would be overwritten by checkout:
        feature.py
Please commit your changes or stash them before you switch branches.
Aborting
```

Description: This error happens when you try to switch branch, but there are changes, which not committed, in your current branch.

Solution 1 - stash

git stash save "<message>" # all changes store in a local stack now and your branch is clean.
git switch <other branch>
git switch <branch> # after finished with other branch – go back to the original branch
git stash list
git stash apply stash@{<index>}
git stash drop stash@{<index>}

Solution 2 – commit WIP.

git add . # staged all the modified files.
git commit -m 'WIP'
git switch <other branch>
git switch <branch> # after finished with another branch – go back to the original branch.
git log -n 3 # you can see WIP commit is the last commit.
git reset HEAD~1 (take you backward 1 commit to where you stop when you switch branch)

Git workflow.

1. Create a new branch for any feature/bug task.
2. Fix/do the feature/bug in your local directory.
3. Stage the files (add).
4. Commit the changes (commit).
5. Switch to master branch (switch).
6. Validate you up to date with remote repo master branch (pull).
7. Merge with local branch master.
8. Handle conflicts .
9. Update remote repo (push).
10. Delete bug/feature branch.