Git Lab – Working in Teams

Objective

The objective of this lab is to become familiar with using git and a remote repository system. The

remote repository system described in the lab is BitBucket. If you wish to use a different one such as GitHub, you may need to modify some of the steps.

In particular this lab covers:

• Creating feature branches for new work

• Creating a pull requests to merge work into “upstream” branches

• If there are merge conflicts you will need to merge manually e.g. using the gitbash command

line

Steps

1. Note you will work in teams but with a SINGLE repository. You will share access to this repository.

2. One team member create a SINGLE Bitbucket repository, make it public so that your team and the instructors can all browse it and READ from it. Include a README when you create the project. This means the project will have a single file in it when it’s created.

3. A public project can be READ by anyone, however your team will also need WRITE or ADMIN access to the project so that they can make code changes.

4. Give your team all have write permission to the repository e.g. On your bitbucket repository web page go to: Repository Settings -> User and group access -> Add members

   You will need to add the email address that each team member uses for bitbucket.

   Each team member will then get an “invitation” email notifying them that they have WRITE access to the repository.

5. **Each** team member will make some changes to the README over the next steps. So **EACH** team member should clone the repository to their own machine using “git clone…” (the bitbucket page will show the clone command so you can copy it).

   For example (your URL will be different):

   git clone https://bitbucket.org/teammember1/exercise-repo.git

6. **One** team member should make a change to the README file with any simple text change e.g. add a message in it with your name. This team member should now make this change available to the rest of the team with “add, commit, push” to get this change to the remote (bitbucket) repository.

7. Verify that you can all see the change in bitbucket, look under the source and commits tabs to see the change.

8. **Each** team member should now run 'git pull' so that they retrieve this new change from the bitbucket repository**.**

9. Now in turn **Each** team member should now make their own unique change or changes to the README file. Add, commit and push your change so that it's available to the rest of your team. AND each other team member should use 'git pull' to bring that change to their local copy of the repository.

10. Repeat step 9 until a change from every team member has been added to the file.

In the above steps we have see every team member add a change **TO THE SAME BRANCH** of a repository. While this is a useful first step in using remote repositories, in production scenarios we would use new branches for different changes (we would not all commit our changes to the "master" branch).

Optional – Advanced Steps – Using branches

IF your team understands the above steps, then try repeating them, however in this case when each team member makes their change in Step 6. They should this time create a new branch first E.g.

1. Create a new branch for your particular change:

   git checkout -b teammember1_branch

2. <make your change to the README file>

3. Add your changes to the repository

   git add .

4. Commit your changes to the repository

   git commit -m "Adding a test change to README"

5. Push your changes to bitbucket BUT now going to the new branch

   git push -u origin teammember1_branch

In the commands above, we have created a new branch, made a change on it and pushed it to bitbucket so that new branch is available to the team.

Each time a team member completes the steps above, you should then bring these changes from the new branch over to the master branch. Do this by creating a "Pull Request" on the bitbucket web screen. This pull request will be from "teammember1_branch" to "master" the branch changes are coming from and the branch changes are going to.

When the pull request has been created, you should then “Merge” the pull request to complete the process.

When the pull request has been “Merged” then every team member should run ‘git pull’ on their own machine to take these changes down to their local copy of the master branch.

The next team member can then repeat the process of creating a branch, committing changes to it, pushing it to bitbucket AND merging it by creating a Pull Request.

The above is a more realistic workflow for Teamwork with Git.