**Setting up git:**

* Install git
* Clone the repository using the following command:

**$git clone** [**https://github.com/amosproj/** **amos2022ss02-audit-chain.git**](https://github.com/amosproj/%20amos2022ss02-audit-chain.git)

Output:

Text

Description automatically generated

* Go to the amos2022ss02-audit-chain/ directory using the following command:

**$cd amos2022ss02-audit-chain**

**Text

Description automatically generated**

* To list all the remote branches:

**$git branch -r**

**Text

Description automatically generated**

**Branching Strategy:**

**Let’s create a separate branch for the new feature and once you are done working on, merge it to the dev only.**

Create a new branch called ＜branch\_name＞. This does *not* check out the new branch.

**$git branch <branch\_name>**

List all of the branches in your local repository. This is synonymous with git branch --list.

**$git branch**

Creating and checkout on the new branch in one line:

**$git checkout –b <branch\_name>**

Text

Description automatically generated

**Naming convention for a branch:**

**branch\_name should look like something like this:**

#type/feature-name

#type : type can be feature, bug or a test

**Ex: feature/consumer-dummy**

**bug/consumer-dummy-bkjwuhwof**

**$git branch –d <branch\_name>**

Delete the specified branch.

**Work on your code/issue!!!**

**To commit and push your code to remote:**

Pushing from your local branch main branch,

To see if you are on which branch

**$git status**

if not, checkout to your branch using:

**$git checkout <branch\_name>**

To add the latest changes to the staging area.

**$git add .**

To commit

**$git commit –m “meaningful commit message”**

In online line:

**$git commit –a –m “meaningful commit message”**

**Ex: $git commit –a –m “#2 Fixed the bug in producer dummy”**

**To push to remote,**

For the first time

**$ git push --set-upstream origin <branch\_name>**

**Ex: $git push –set-upstream origin #2-consumer-dummy**

Second time onwards,

**$git push**

Then, go to GitHub and create a pull request to main branch and add a reviewer if necessary.

**A screenshot of a computer

Description automatically generated with medium confidence**

Click on compare and pull request

**A screenshot of a computer

Description automatically generated with medium confidence**

Before creating the pull request make sure to add description of what you did and tag the issue #issue\_number.

Remember the base should be dev and compare should be your local branch.

**Idea is that release manager will push the code from dev to main after testing on every Tuesday i.e while creating release candidate.**

**Advanced:**

If there are changes to the same file on different branches(should not happen in our case as we work on different branches), then when you try to commit the merge conflict will arise.

$git status : read the messages

To go

Delete the things you want to keep by removing the unwanted lines

Stage it

$git add file\_name

$git status: to check if the conflicts are gone

$git commit

Notes:

Before you start working on something, pull from remote so that local is up to date (this will help minimizing merge conflicts as well)

**$git pull**

If you want to see current status of the local repo

**$git status**

To see the log with all the commits and details:

**$git log**

To see more details on repo:

**$git remote –v**

To see all the branches(local + remote)

**$git branch –a**

To see only remote branches

**$git branch -r**

To see commit history in one line:

**$git log --graph --oneline**

Note: If you have untracked files and git pull doesn’t work

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

To delete all the untracked file(files that are changed locally but not needed anymore)

**$ git clean -d -f**