

MP3 Tips from Course Staff

By Okan K.

Disclaimer: This is my experience and my recommendations; these are not default ways of doing this MP. These are **JUST** recommendations. If you have questions about this document, come to my OH, send me piazza post, or ping me (okan2) on discord.

Before Starting MP3:

- Make sure everyone has Local setup before the MP¹.
- Be confident about using GDB and know how to use gdb via ssh. To get more information about this look at Appendix B or @377.
- Discuss your goals with your team! Make sure everyone agrees on the workload to put into this project. Here are some important questions to discuss with your group:
 - What are you grade are you aiming for this class?
 - What is your goal for the final project? (B,A, or winning the extra credit competition)
 - How much time per week are you willing to put in?
 - Would you like to do extra credit?

Before Starting Any Checkpoint:

- Read the documentation (RTDC) and divide the checkpoint into smaller parts. Then rank the difficulty of each part. Here is an example for Checkpoint 1 from most to least difficult:
 1. Paging
 2. Exception Handler
 3. RTC Interrupts
 4. Keyboard Interrupts
 5. GDT
- Create new branches for each part on GitLab².
- Discuss with your team about your workload for that week.
- Divide the work for each part. Especially for checkpoint 1 and 2, working separately and then merging the parts is recommended.

¹ If you are having issues with Windows target path, look at Appendix A, to learn how to setup a .bat file to run qemu.

² Gitlab tutorial can be found in Appendix C.

Submitting a Checkpoint:

- Discuss and explain each other your code, make sure everyone understand how it is working. This important in two main reasons. Firstly, during the final checkpoint you should be able to answer any question related to the code. Secondly, since every checkpoint will build on top of the other knowing the details of other components are essential. For instance, when you are writing the terminal driver, you should know how the keyboard driver is implemented.
- While going over the code, check for comments and other styling features.
- If you used separated branches make sure, when you finished the entire code is in main branch.

Appendix A: Windows Target Path Issue

In windows, a shortcut target path can have max 256 characters. Depending on where you placed your local setup you might have issues such as qemu not running, or unable to setup ssh. To fix this you can create a .bat script to run the qemu with preferred parameters. This procedure takes only 4 steps.

1. Create a .txt file and copy the following code:

```
0: if not DEFINED IS_MINIMIZED set IS_MINIMIZED=1 && start "" /min "%~dpnx0" %* && exit
1: $(Disk your WFH setup is in)
2: cd "$(Path to your work from home setup) \ece391\qemu_win"
3: qemu-system-i386w.exe -hda $(path to your ece391 share folder)\ece391_share\work\$(Group github folder)\student-2:
3 distrib\mp3.img -m 256 -gdb tcp:127.0.0.1:1234 -S -name mp3
4: exit
```

On line 0, we are telling the command prompt to minimize itself once the script it fully ran. On line 1, we are going into the disk our WFH setup is in. On line 2, we cd into the our qemu_win folder. On line 3, we run qemu with specific parameters such as -gdb, tcp, -name, etc. **Make sure all the parameters are on the same line there should NOT be line break(\n).**

2. Click Save As and save the file as \$(Insert name of the OS).bat
3. Run the script and see if it boots up.

Appendix B: GDB and SSH

1. Add the following line to your target path:

```
-redir tcp:2022::22
```

2. Open cmd, don't use gitbash since it sometimes doesn't work, and ssh into the kernel with following line:

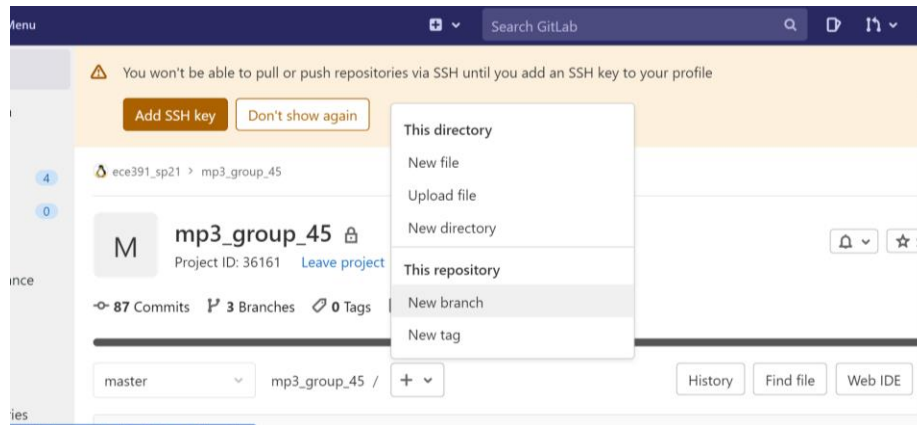
```
ssh user@localhost -p 2022
```

Also for more information about gdb commands you can take a look at this github:
<https://gist.github.com/rkubik/b96c23bd8ed58333de37f2b8cd052c30>

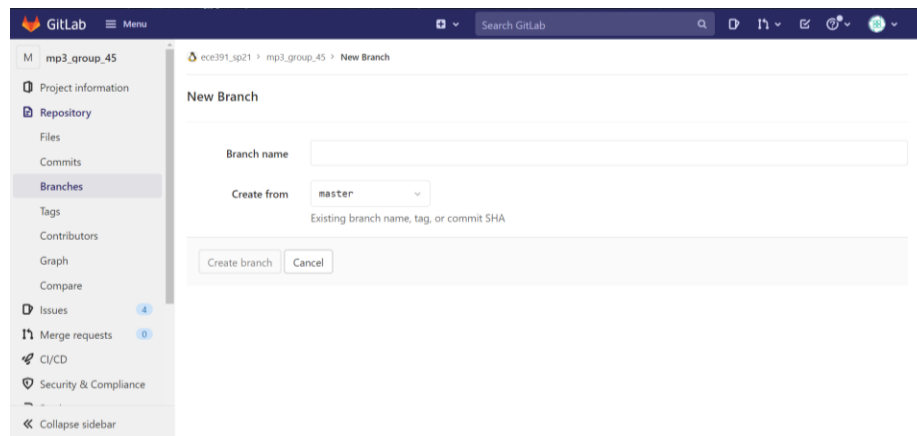
Appendix C: Gitlab Tutorial

Appendix C.1: Creating New Branches:

In your Gitlab click on the + sign and pick New Branch.

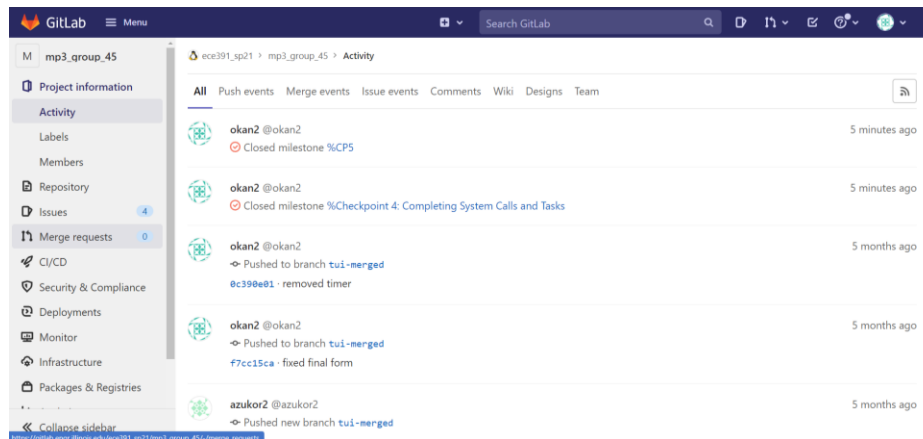


Here fill in the details for the new Branch. Make sure you double check the “create from”. Most of the time you want it to be made from master. However, if you haven’t intergrated a feature to master yet, but need it for another one; you need to pick the branch with that feature in “Create from”. For instance, you have a branch called PIC, and implemented PIC there but didn’t intergrate it to main yet. You want to create IDT branch, when creating it you would select PIC, in create from.

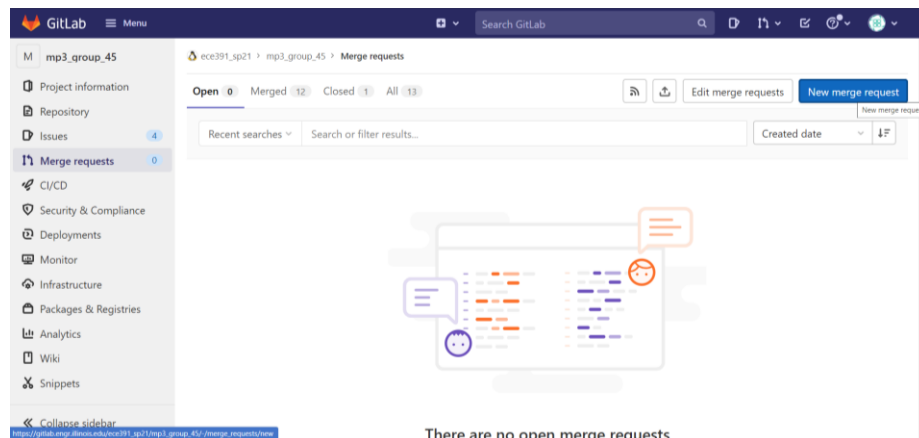


Appendix C.2: Merging Branches:

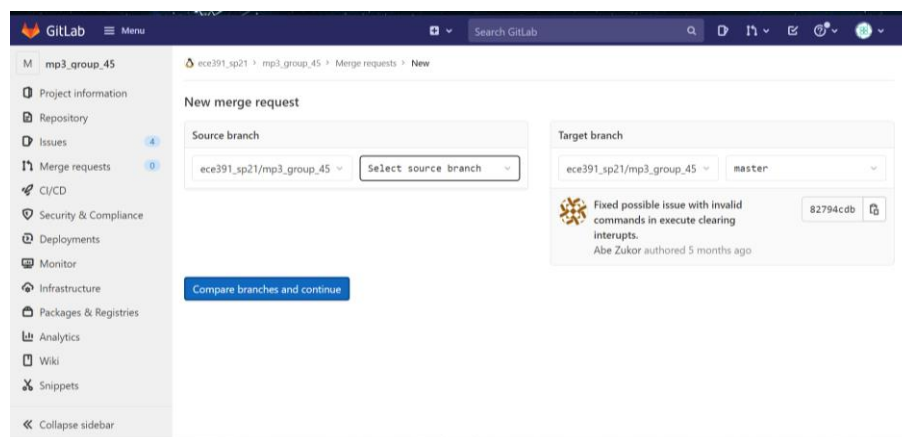
Click on merge requests



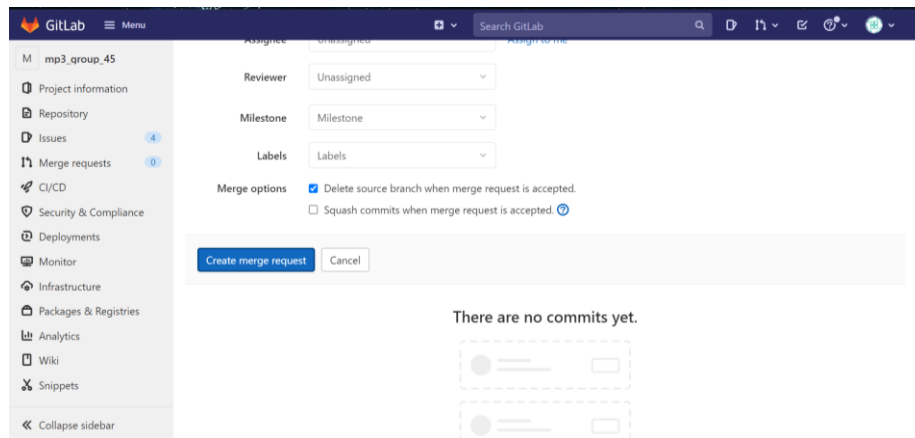
Click on create New merge request.



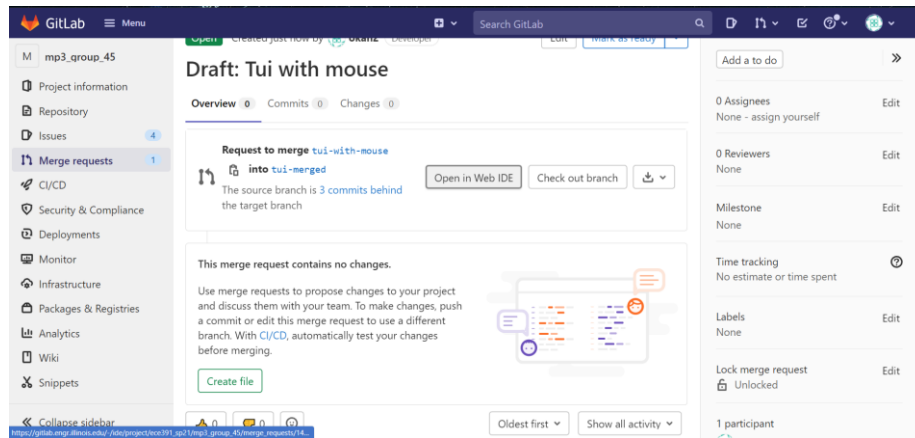
Select the two branches you would like to merge. Source will be the code copied to target. Usually you want target to be master or “create from” branch.



Create your merge request. You can assign reviewers if you want.



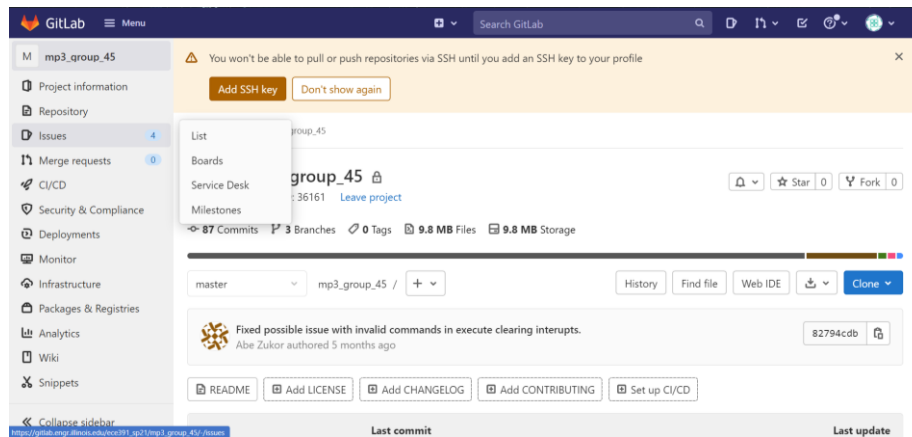
After creating the merge request. You need to resolve merge conflicts.



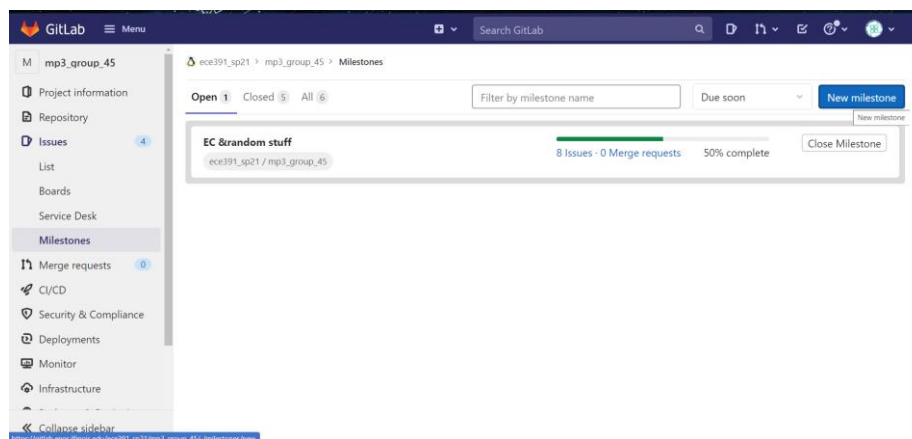
And it will tell you which files are changed and ask you which part to keep/ move/ remove.

Appendix C.3: Milestones

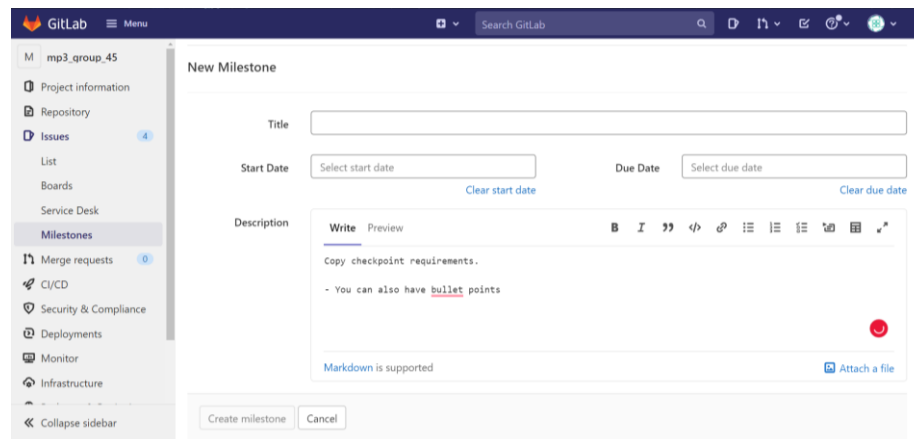
I recommend you set your Milestones as Checkpoints. To create a Milestone hover over issues.



Click on “New Milestone”, to create new milestone.



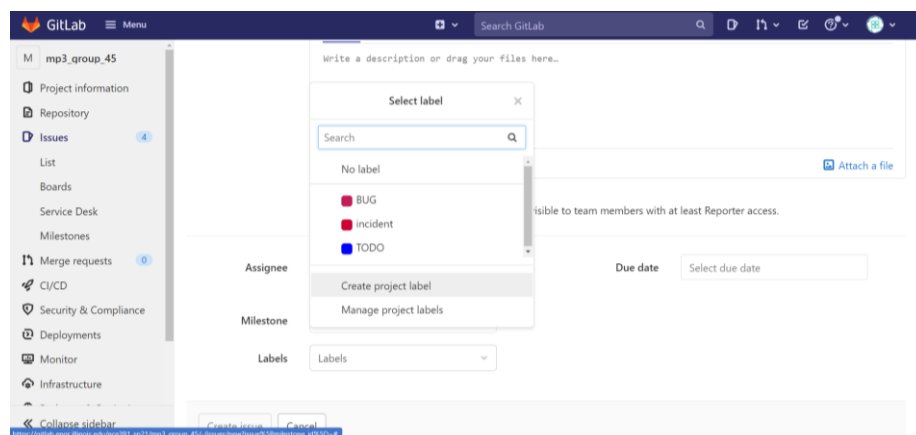
Fill in the information for the milestone, add the start date and ,end date for checkpoint. I recommend making the title the Checkpoint.

A screenshot of the GitLab web interface showing the 'New Milestone' form. The left sidebar contains navigation links for Project information, Repository, Issues (with a count of 4), List, Boards, Service Desk, Milestones, Merge requests (with a count of 0), CI/CD, Security & Compliance, Deployments, Monitor, and Infrastructure. The main form has fields for Title, Start Date (with a 'Clear start date' link), Due Date (with a 'Clear due date' link), and a Description field with a rich text editor. The description contains the text 'Copy checkpoint requirements.' and '- You can also have bullet points'. There are 'Create milestone' and 'Cancel' buttons at the bottom.

Appendix C.4: Bug Log:

I personally recommend creating 2 labels: BUG and TODO in your gitlab issues as following:

1. Try creating a new “Issue”
2. Click on Label dropdown menu, add click “Create Project Label”.

A screenshot of the GitLab web interface showing the 'New Issue' form. The 'Select label' dropdown menu is open, displaying options: 'No label', 'BUG' (with a red square icon), 'Incident' (with a red circle icon), and 'TODO' (with a blue square icon). Below these are links for 'Create project label' and 'Manage project labels'. The background form shows fields for Assignee, Milestone, and Labels (a dropdown menu). The URL at the bottom is 'https://gitlab-engr.devs.com/issues/1_group_45/issues/new/issue%3Dmilestone_d9%3D-8'.

After this whenever you create a new milestone and add these bugs/ todos it will be linked to that milestone, making debugging easier later. To link a milestone, when you are creating a gitlab issue you can click on Milestone dropdown and pick the Milestone you want.