Arūnas Butkus EKSfmu-16

**Practical Task 2**

What I did:

**Part 1:** “Repository creation, adding files into repository”

The 7 steps here are not as clear as having 7 steps would imply, but from what I gather at this point I should have two repos, which are the same, “server\_side” is the origin one where the stuff comes from and goes to; and “client\_side” – simulating some user getting the original repo. I start of by following YouTube tutorial of setting up git SSH server and client (https://youtu.be/lXSZUuDW4nY)(Fig 1)(What is shown in figure was wrong, missing a line). Sometime later: networking is hard! Even following step by step, something breaks aaaand now I don’t know what to do anymore. Give up for now…

Attempt number two. Trying out git options available on Windows. Checked out “GitHub Desktop” - simple install with a simple new repo creation (Fig. 2). Committing is simple and obvious (Fig. 3). To emulate the harder way of using git with bash – installed Git-2.14.1-64-bit – git for windows command line. Installation is more involved but has plenty of explanations in it (Fig 4-9). Now having CMD access to git I check configuration file and I see a bunch of “GitHub Desktop" configurations with ones I added through CMD (user and email), as I did with bash, at the end (Fig. 10).

So considering only the part description I’ve done it in multiple ways already, but the steps also asks to connect two git users to one repo and commit files from the client one. Then now I will attempt to connect windows user to the repo in VM. Or vice versa.

Hour, or few, later I’m back! And it works. I won’t explain what is already explained in the before mentioned YouTube video as I once again followed that one. The issue was I did not notice that that interface file based static IP setting actually prevented internet access, even though it allowed to see all computers connected to the router. This caused issues getting openssh-server and git client on new VM. Circumvented that issue by setting static IP for the gitServer VM in the router configuration. By the way, now I’m running two VMs: from laboratory work 1 as a client and, downscaled in terms of hardware use, “gitServer” VM. Proof of reaching the end of that tutorial is in Figure 11.

Thus client can now push and pull data from gitServer. Wonder if ”GitHub Desktop” can clone that repo (Fig 12). And the answer is no due to authentication issues that I have no clue how to solve (Fig 12). First idea was, since I added SSH authentication, to use PuTTY to open SSH connection to get to the gitServer. Generated an SSH key with PuTTY key generator (Fig. 13), authorized that key as git user in the gitServer, allowed git users to use bash shell, and voilà (Fig. 14). But did not know what to do with that…

So then I noticed that installing git command line software added git bash to the right click menu (Fig. 15) (already forgetting what I did in Figure 4). Using half-baked knowledge gathered till now, went and made a folder where I’d like to store my gitServer repo. Opened git bash in it, executed git init --bare which should’ve given an empty git repo, but instead made a mess of folders and files (Fig. 16). So then proceeded to try pulling remote repo. Don’t quite remember at which point, but eventually it worked; after adding remote origin to git settings (Fig. 17) and when I figured out what parameters git pull requires. Did some cleanup with .gitignore file on Windows side repo and eventually got some decent repo on Windows side too (Fig. 18-20).

Just to finally cement the git command line/bash way of data sharing, made a change to the text.txt file in Windows (Fig. 21), committed and pushed it to the gitServer (Fig. 22), and pulled it on Ubuntu client (Fig. 23). Committing being the hardest part since I did not specify -m parameter with a message which launched VI editor which then required me to go through three more YouTube tutorials where the least Hindu person of those 3 finally told me how to save and close that editor… ESC -> Shift+zz (facepalm)

**Part 2:** “Getting files from repository, making changes”

I did it for the most part already, but I’ll do it again with the provided text then (Fig. 24). Mostly.

**Part 3:** “Conflict resolving”

First I pulled the change from previous part (Fig. 25). Then I edited and pushed the changes similar to as instructed (Fig. 26). And then I did as instructed on the Windows side (Fig. 27), which ended up with prevented push due to files in repo being newer than local pull. So conflict now appears when trying git pull. (Fig. 28). Resolved the conflict by removing the auto-generated lines 3, 5 and 7 and re-pushing (Fig. 29).

Considering the experiences so far: Turquoise SVN is better; though I didn’t have to configure it and was just using as a client…

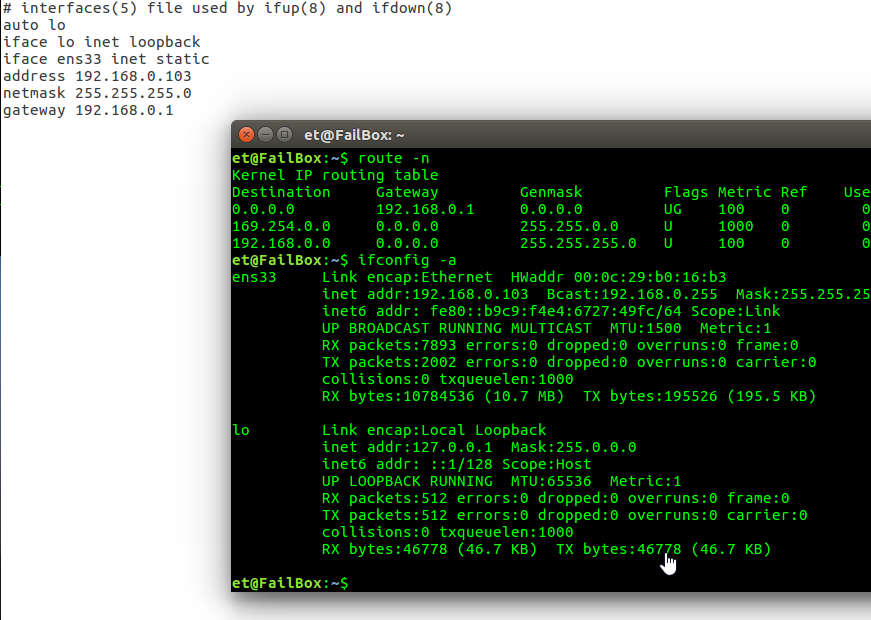


Fig1

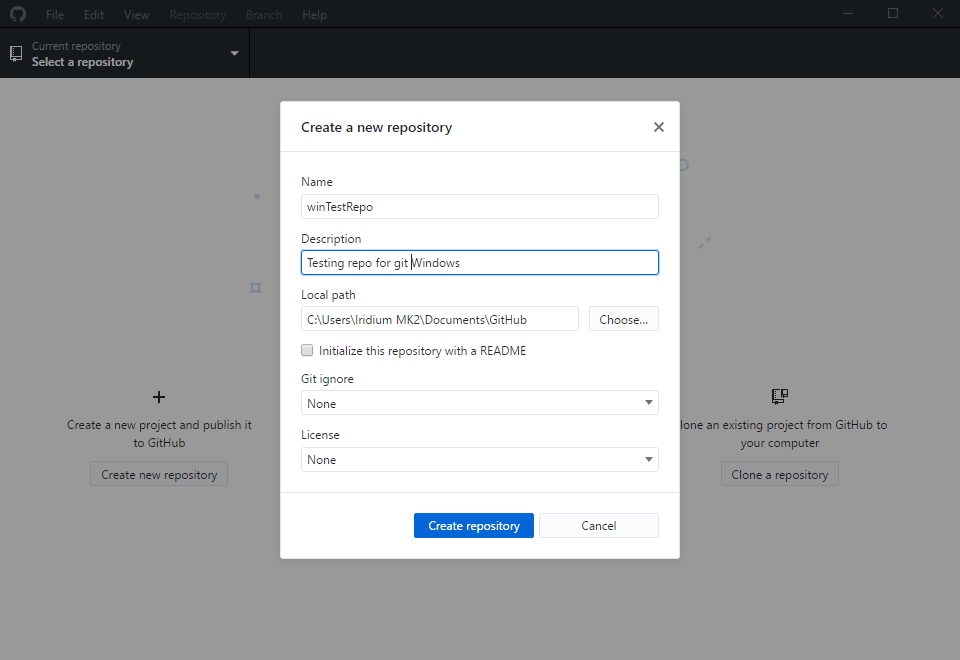


Fig2

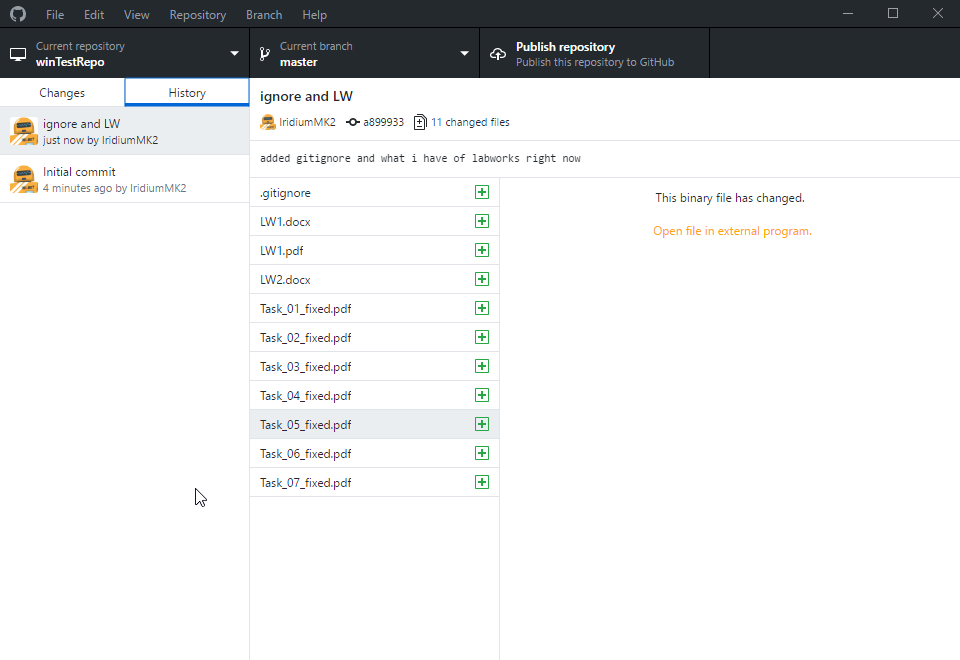


Fig 3

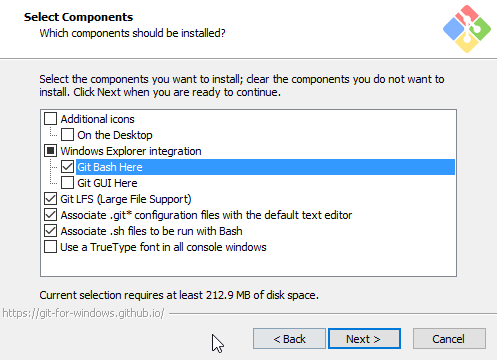


Fig4

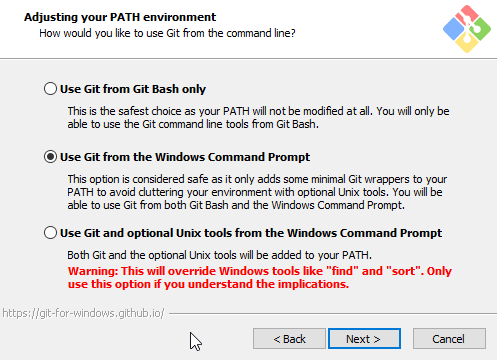


Fig5

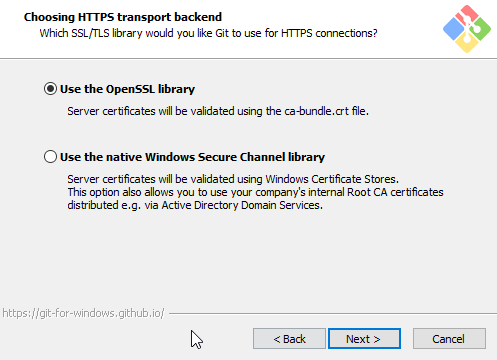


Fig 6

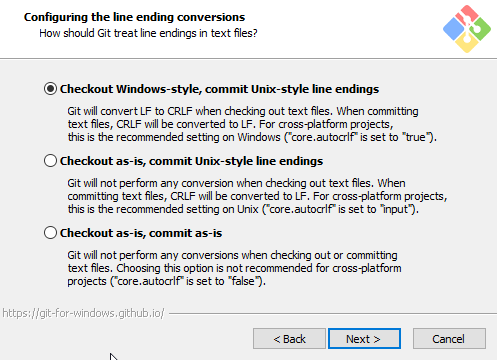


Fig 7

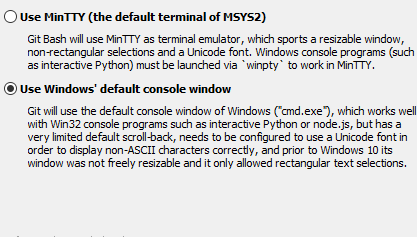


Fig 8

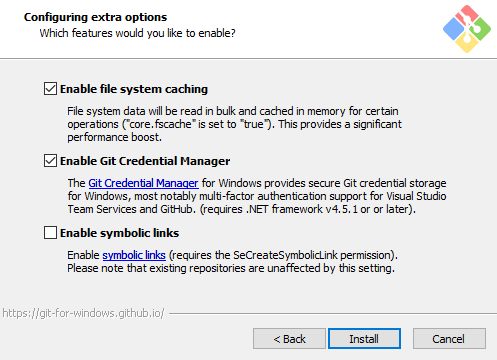


Fig 9

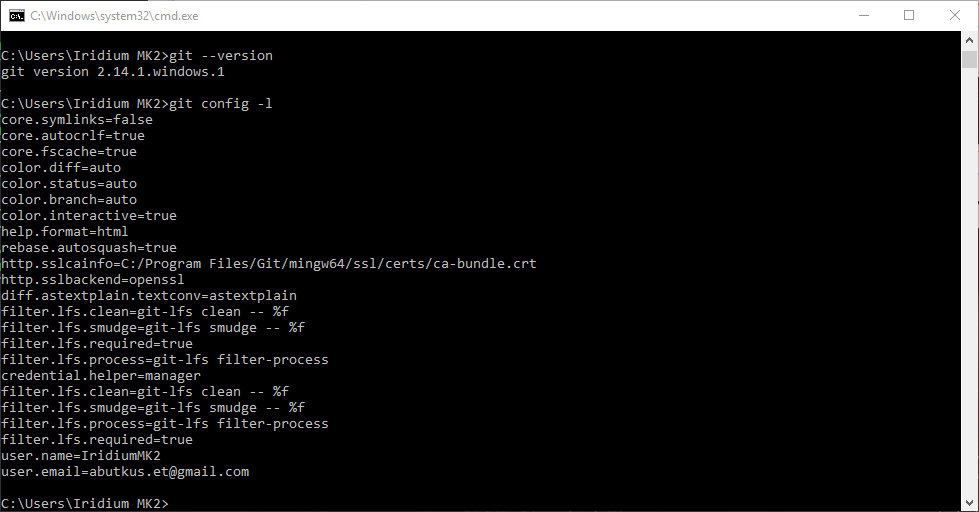


fig10

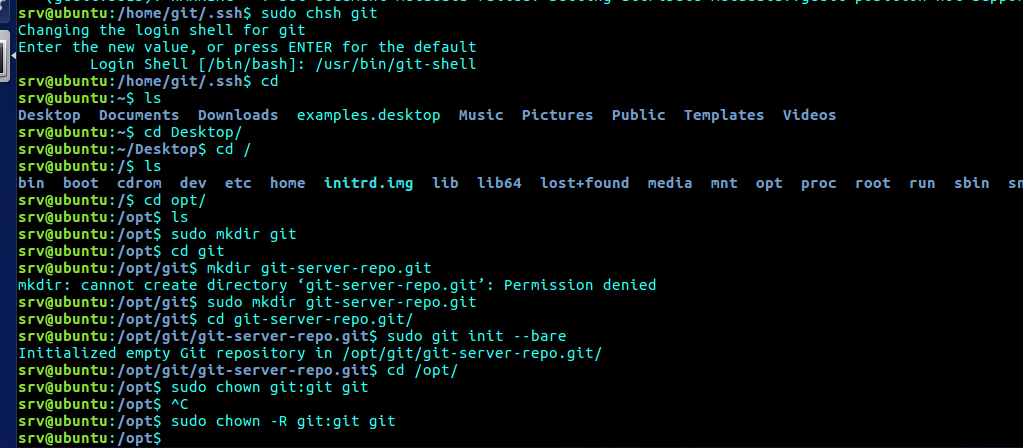


Fig 11

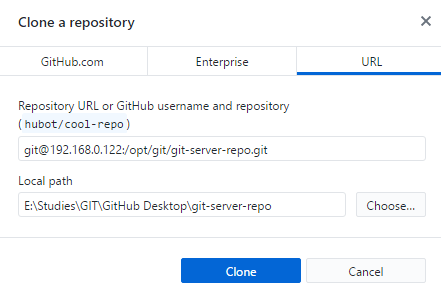


Fig 12

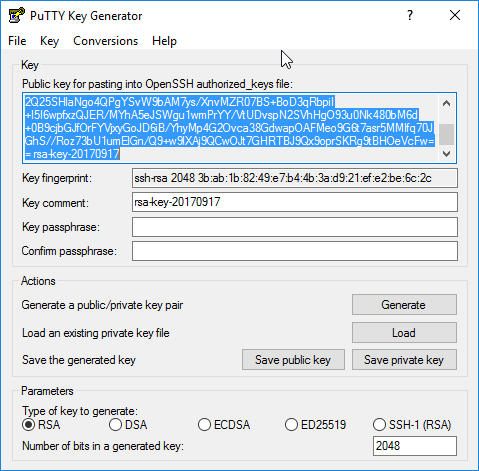


Fig 13

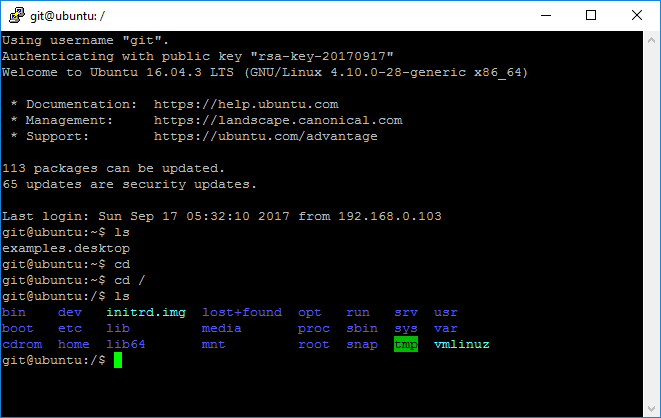


Fig 14

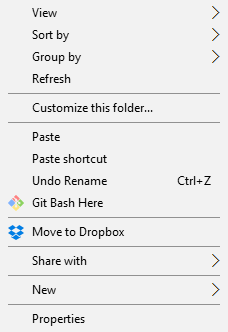


Fig 15

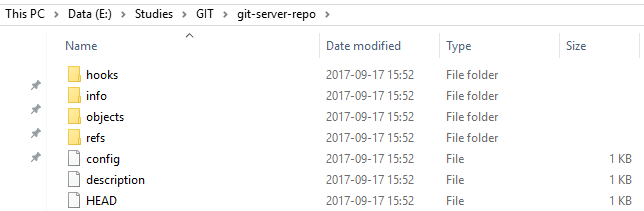


Fig 16

C:\Users\Iridium MK2\Documents\ShareX\Screenshots\2017-09\ApplicationFrameHost_2017-09-17_16-41-17.png

Fig 17

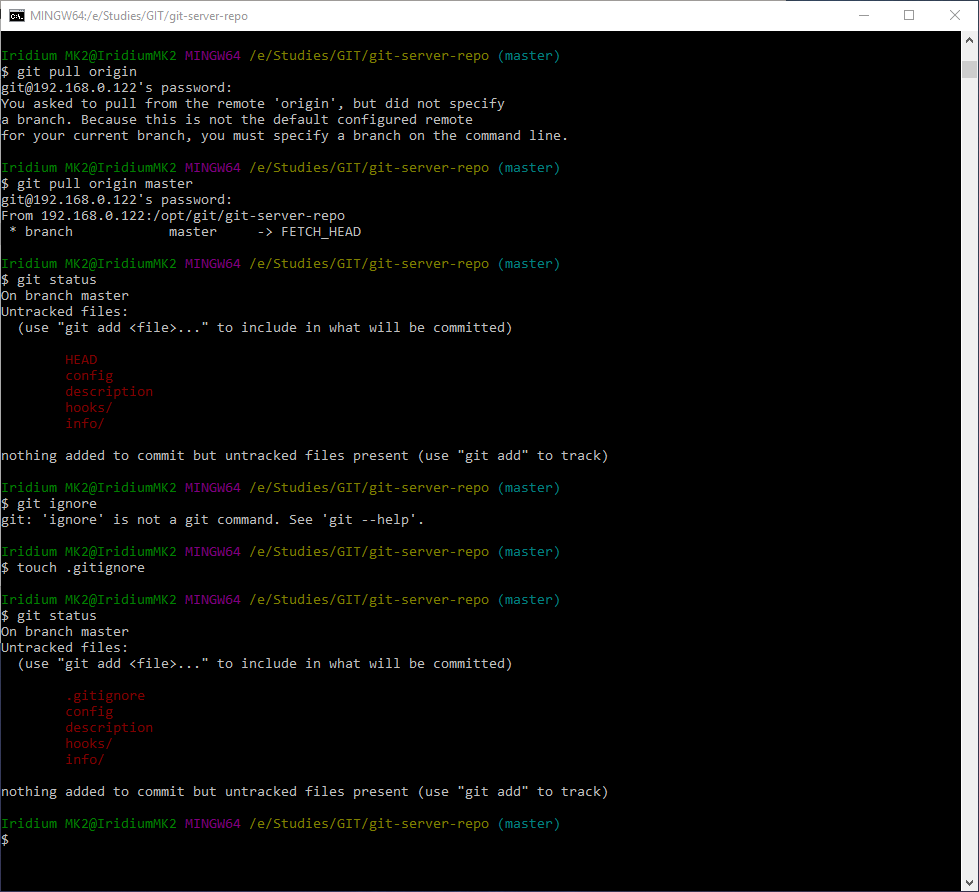


Fig 18

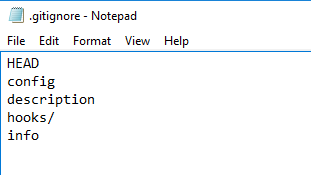


Fig 19

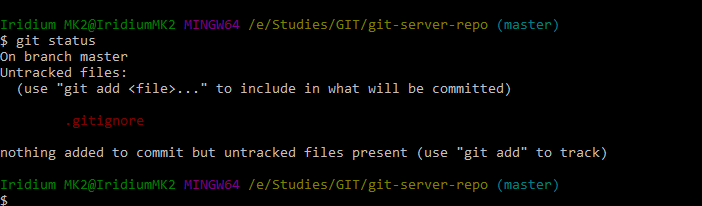


Fig 20

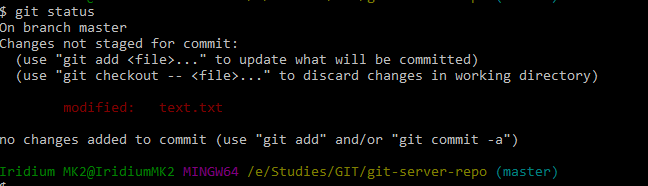


Fig 21

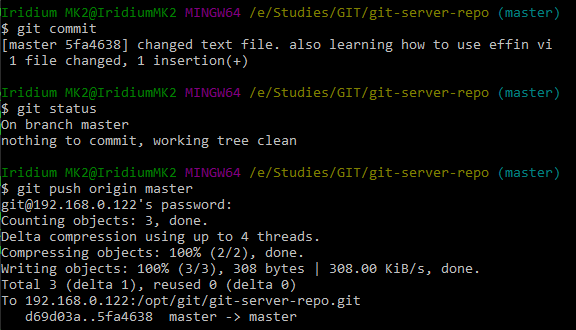


Fig 22

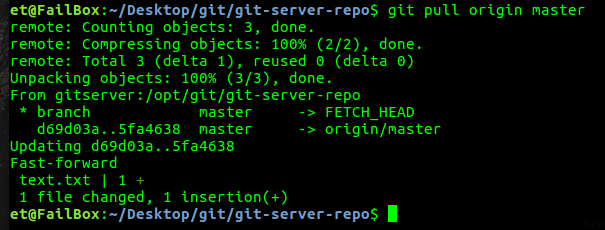


Fig 23

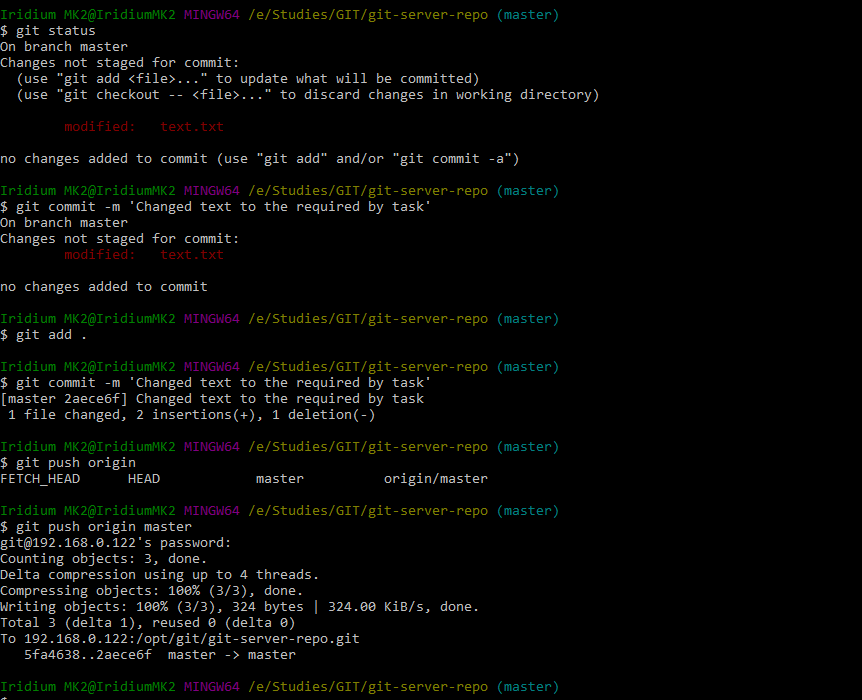


Fig 24

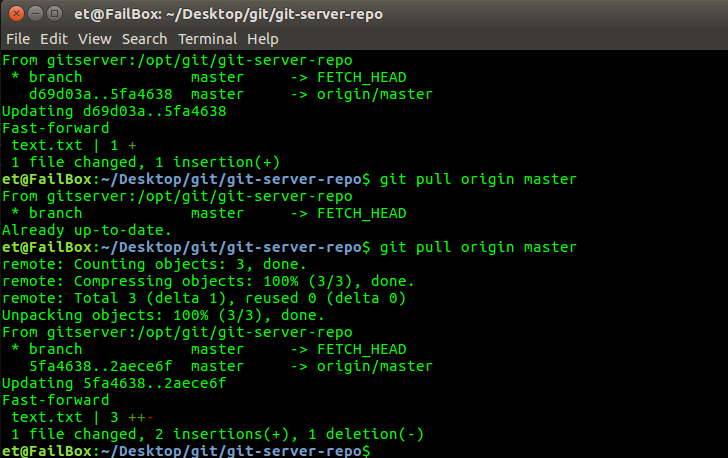


Fig 25

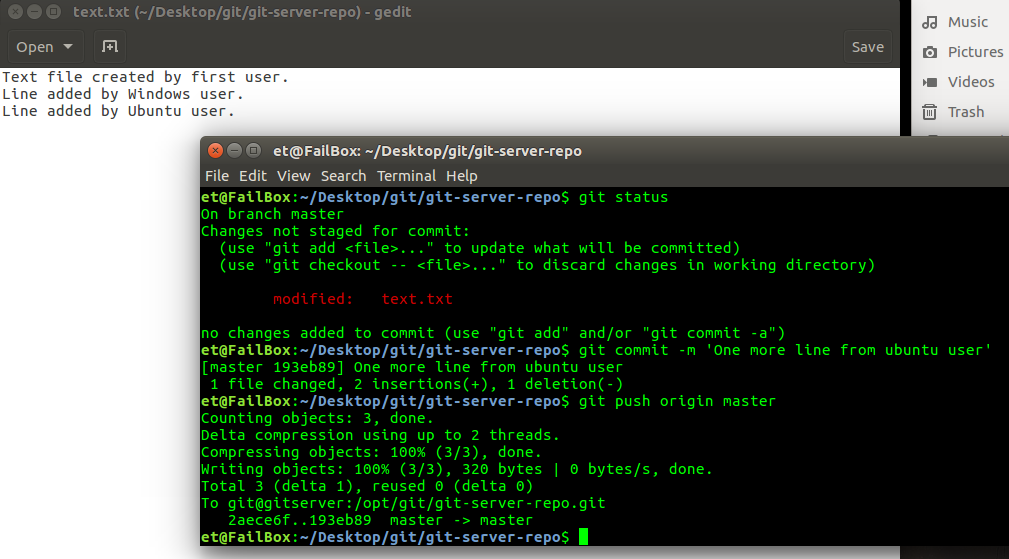


Fig 26

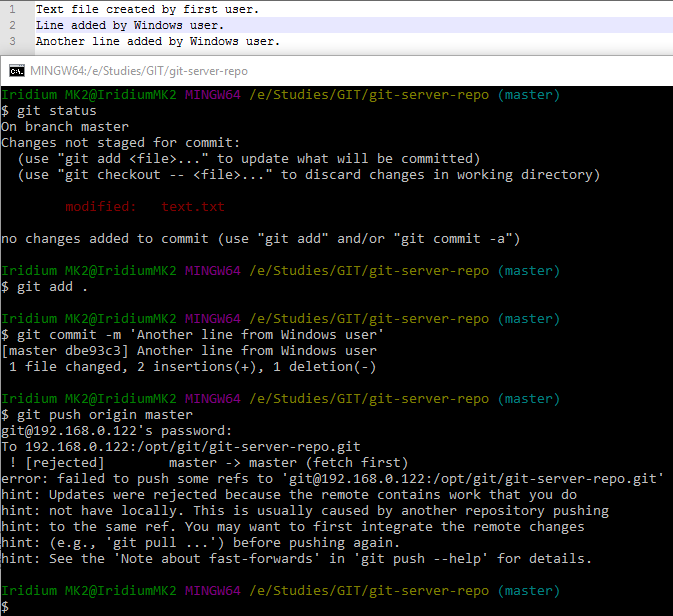


Fig 27

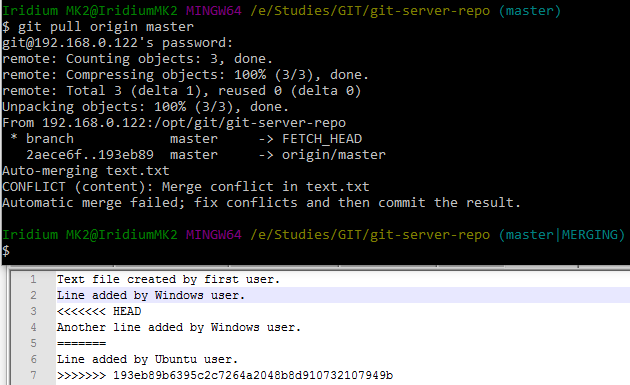


Fig 28

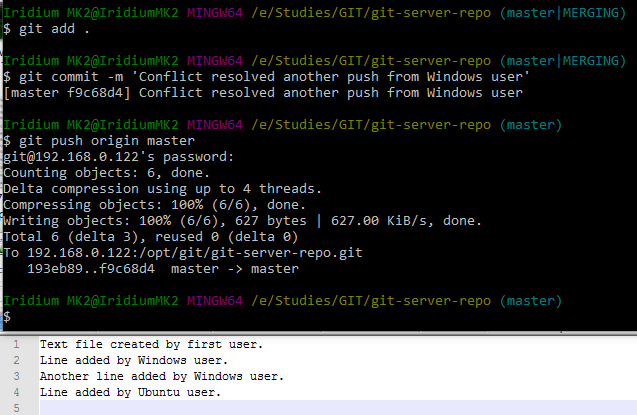


Fig. 29