

Last edited by  [Mohammad Kamrul Hasan](#) 1 year ago

Git & Gitlab

- [Steps to get started with GitLab](#)
 - [Installing Git](#)
 - [Configuring Git](#)
 - [Starting with any project](#)
 - [Cloning the project](#)
 - [Cloning with SSH \(recommended\)](#)
 - [Cloning with HTTPS](#)
- [Branching & Merging Workflow](#)
 - [Branching](#)
 - [Creating a new branch](#)
 - [Branch naming strategy](#)
 - [Merging](#)
 - [Checklist before creating a merge request](#)
 - [Git add, commit & push steps](#)
 - [Follow the Git Commit Guideline](#)
 - [Creating a new Merge request](#)
 - [Checklist after creating a merge request](#)
 - [Responding to MR response:](#)
 - [Replying to MR Comments:](#)
 - [Reviewing a Merge request](#)
- [Git commands for developers](#)
- [Resources](#)

Steps to get started with GitLab

• Installing Git

- To install Git , open the terminal and write the following commands:

```
$ sudo apt-get update

$ sudo apt-get install git
```

- After installation, to verify that Git is successfully installed, write the following command:

```
$ git --version
```

• Configuring Git

- Configure your Git username and email using the following commands:

```
$ git config --global user.name "your_username_here"

$ git config --global user.email "your_email_here"
```

• Starting with any project

◦ Cloning the project

▪ Cloning with SSH (recommended)

- SSH stands for Secure Shell. The SSH protocol provides security and allows you to authenticate to the GitLab remote server without supplying your username or password each time. For a more detailed explanation of how the SSH protocol works, we advise you to read this: <https://www.digitalocean.com/community/tutorials/understanding-the-ssh-encryption-and-connection-process>
- To clone with SSH, we need to generate an SSH key. When we generate an SSH key, we get a public key and a private key. Then, the public key is added to the GitLab server to use SSH key authentication for access control.
- Before generating a new SSH key pair, let's check if your system already has one at the default location. Run the following command:

```
$ cat ~/.ssh/id_rsa.pub
```

- If you see a string starting with `ssh-rsa`, you already have an SSH key pair and you can skip the generating SSH key pair portion.

```
marjan@marjan-pc: ~/Documents/bpm  ×      marjan@marjan-pc: ~  ×  ∨
marjan@marjan-pc:~/Documents/bpm$ cd
marjan@marjan-pc:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCfg70aCaOMCcsKaNaLwQCXiFTD1AMsp+ATs0F+xbRO
E7rbLNkgRD/LoJ3wnr1bB9UXcE/+Z+0Dmrs5dCpfBnieUsvPCJwa03fZtWJNDust+gjs7Y0tz7YYThSD
WVRdLiRMLxUmwXv857T+pxOrVR7yXWLHxei6DiycE8pINw0TS0m3Msvr0ZwKvXLq5JPdK1YdfWqTae17
qk0n3WqsF0Ga/TsprXXiLRKWL0wB4BSSF0Nubz+a8pwjm8PwzQsMJ784EmydCw/OjNqwuIrUSCZUeMhI
Fx63+iznm05Zkq6gizlkJc0G0lVCE5xxG6oJ00DBsvZDI0//OpWNUQ3/G/R+6nr6F4ABRb/a3cTfY9M
htL7TFuvKGiza5mKYN5hRteyQ3UaCJwnz8Uon5NR9xpK854DnhVjNrRRIfRgxEwdQ/Uq3dDdgZCFRK30
J71yTxx6I+has5XQfaDEaZ58hanVVhcU6finK7AID7raxk9ioLtA2aJdI10s581zjBIg9Ks= marjan@
marjan-pc
marjan@marjan-pc:~$
```

- In case your system doesn't have SSH key pair already generated, you will find the following result:

```
+      ahmed-saquib@jarvis: ~  🔍  ⋮  -  □  ✖
ahmed-saquib@jarvis:~$ cat ~/.ssh/id_rsa.pub
cat: /home/ahmed-saquib/.ssh/id_rsa.pub: No such file or directory
ahmed-saquib@jarvis:~$
```

- To generate the SSH key pair, We will be using a tool called `ssh-keygen`. Open the terminal and run the following command:

```
ssh-keygen -t rsa -b 4096 -C "office-pc"
```

- Then, you will be prompted to input a file path to save your SSH key pair to. Default path is `/home/marjan/.ssh`. Leave it as it is and hit the Enter key.

```
marjan@marjan-pc:~$ ssh-keygen -t rsa -b 4096 -C "office-pc"
Generating public/private rsa key pair.

Enter file in which to save the key (/home/marjan/.ssh/id_rsa): Created director
y '/home/marjan/.ssh'.
Enter passphrase (empty for no passphrase):
```

- Then you will be prompted to input a password to secure your SSH key pair.
- Finally, your SSH key pair has been generated.

```
The key's randomart image is:
+---[RSA 3072]---+
|      . . .      |
|      . o      . |
|      . o +.      o |
|      + . . +. *   |
|      . . +So. *+. o . |
|      . . + . . = = o |
|      . E . o = *   |
|      . o + + B +   |
|      . . o B = +   |
+-----[SHA256]-----+
hasan@hasan-pc:~$
```

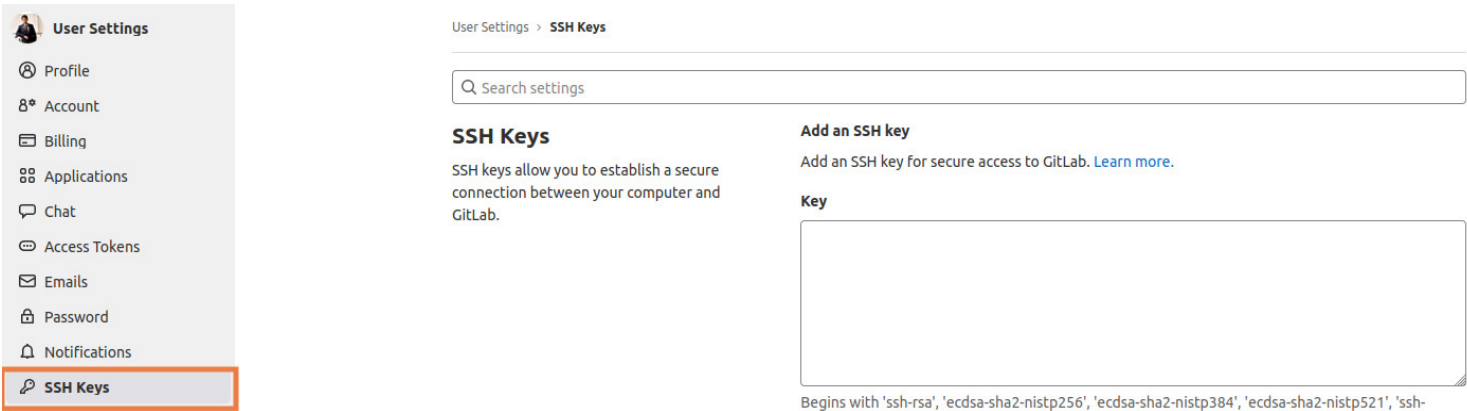
- The next step is to add your SSH public key to `GitLab`.
- To get the SSH public key, write the following command:

```
$ cat ~/.ssh/id_rsa.pub
```

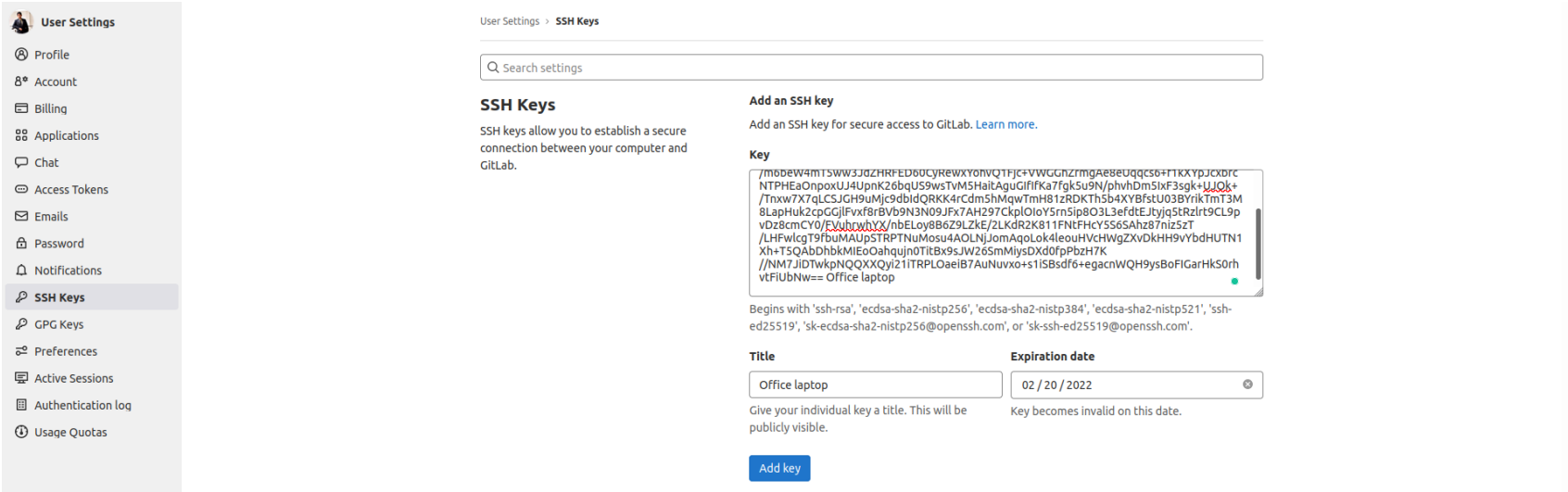
example:

```
marjan@marjan-pc: ~/Documents/bpm
marjan@marjan-pc:~/Documents/bpm$ cd
marjan@marjan-pc:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCfg70aCaOMCcsKaNaLwQCXiFTD1AMsp+ATsOF+xbRO
E7rbLNKgRD/LoJ3wnr1bB9UXcE/+Z+ODmrs5dCpfBnieUsvPCJwa03fZtWJNDust+gjs7Y0tz7YYThSD
WVRdLiRMLxUmwXv857T+px0rVR7yXWLHxei6DiycE8pINw0TS0m3Msvr0ZwKvXLq5JPdK1YdfWqTael7
qk0n3WqsF0Ga/TsprXXiLRKWL0wB4BSSF0Nubz+a8pwjm8PwzQsMJ784EmydCw/OjNqwuIrUSCZUeMhI
Fx63+iznm05Zkq6gizlkJc0G0LVCE5xxG6oJ00DBsvZDI0//OpWNUQ3/G/R+6nr6F4ABRb/a3cTfY9M
htL7TFuvKGiza5mKYN5hRteyQ3UaCJwnz8Uon5NR9xpK854DnhVjNRRifRgxEwdQ/Uq3dDdgZCFRK30
J71yTxx6I+has5XQfaDEaZ58hanVVhcU6fInK7AID7raxk9i0LtA2aJdI10s581zjBIg9Ks= marjan@
marjan-pc
marjan@marjan-pc:~$
```

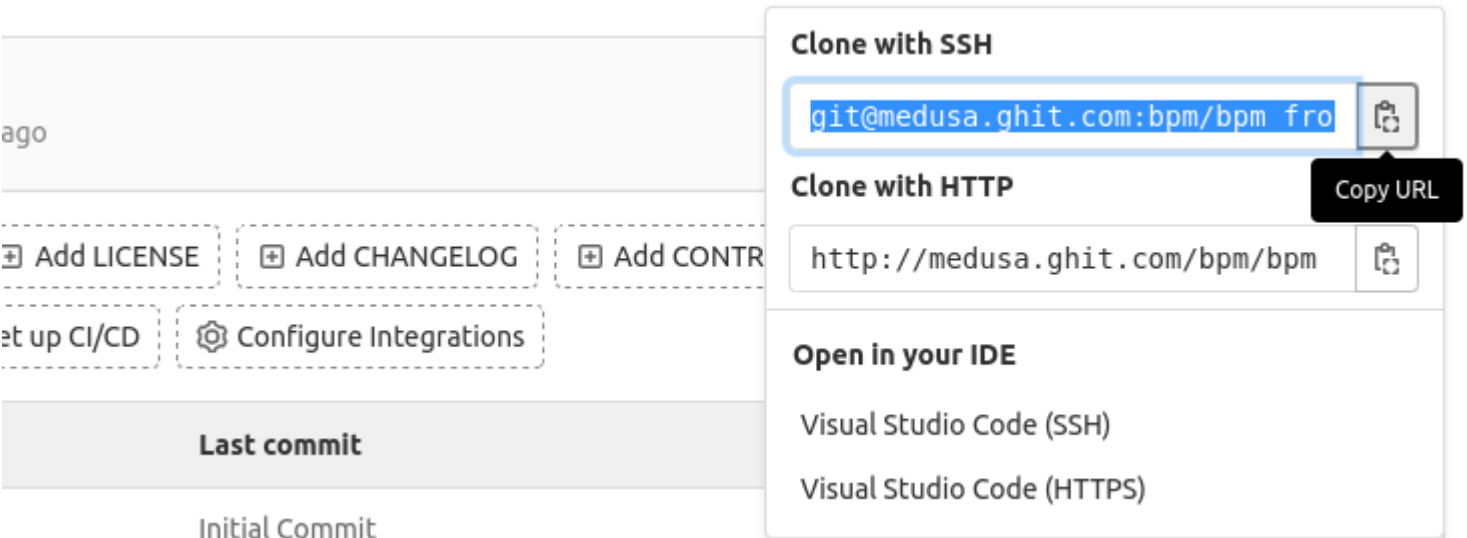
- Copy the SSH key. Then go to `edit profile` and click on the `SSH keys` tab on the left side of the menu.



- Paste the key and add an expiration date for that key.



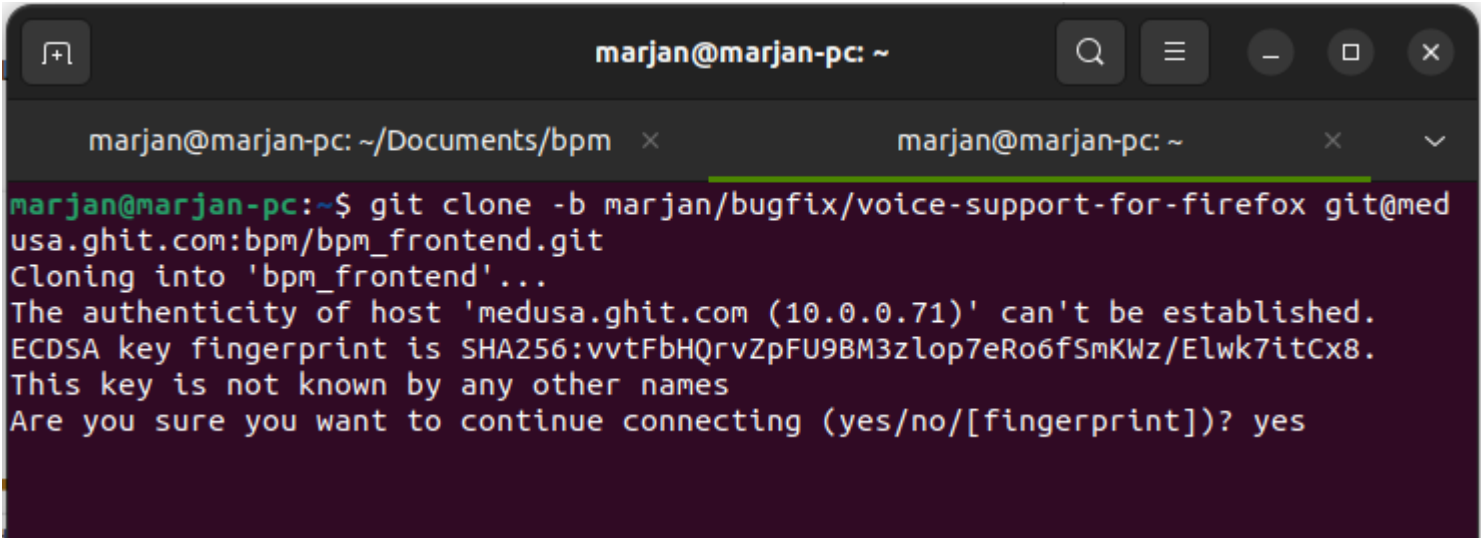
- Now, you have to open the project from <https://pms.ghitbd.net/>. And copy the URL:



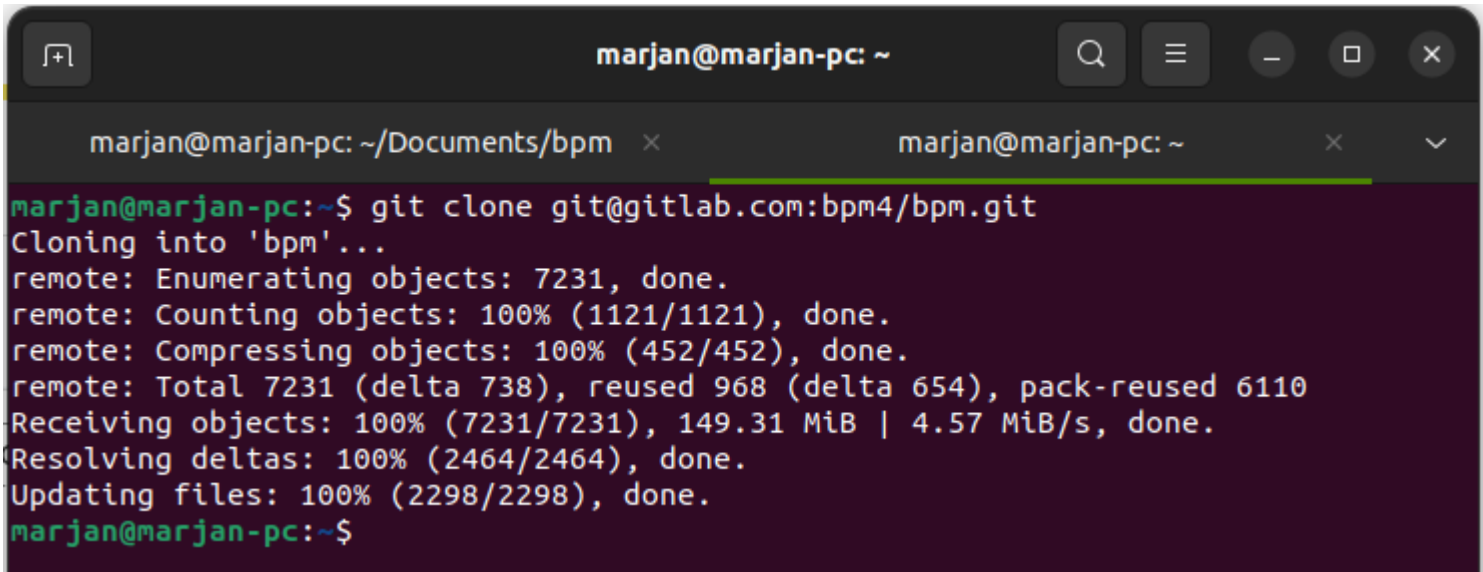
- After that, go to the directory where you want to clone the project. Open the terminal and write the following command:

```
$ git clone -b "your_branch_name" paste_the_URL_here
```

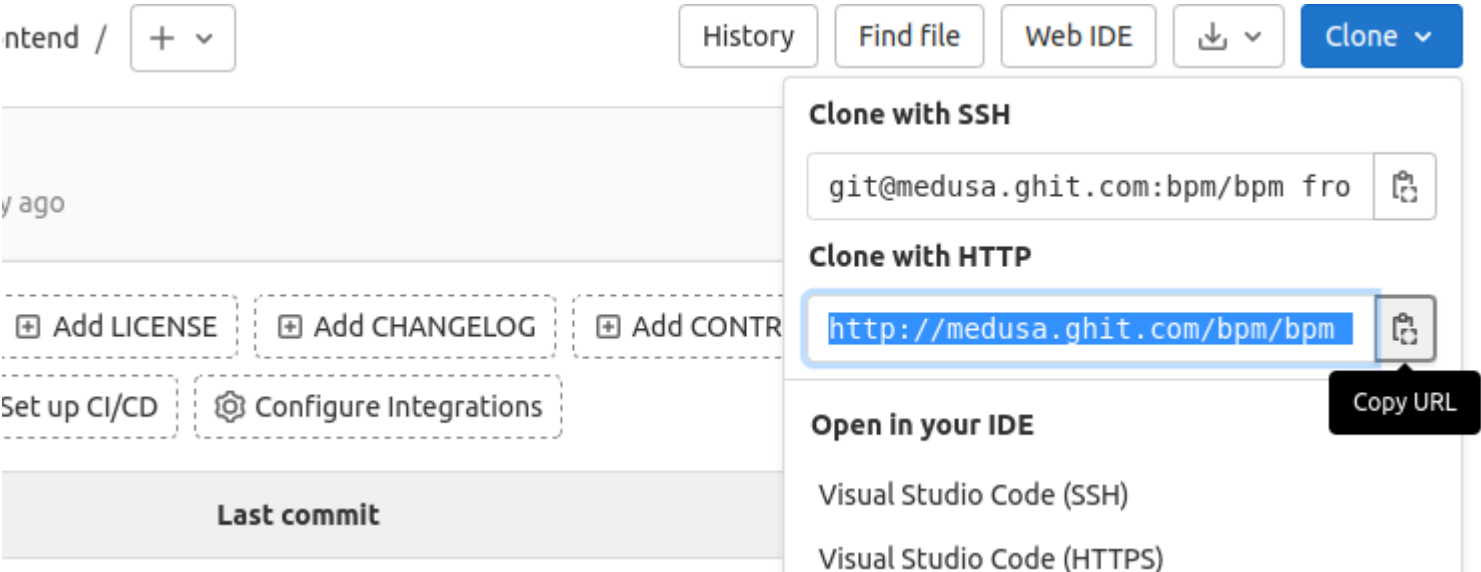
- After that, you will be asked if you want to continue connecting. type `yes` .



- Then one pop-up will come, asking you to enter the password which you have given while generating SSH key pair. After you give the password and press `Unlock` , You will find a new folder named after the project name.



- **Cloning with HTTPS**
 - open the project from <https://pms.ghitbd.net/>. Then copy the URL:

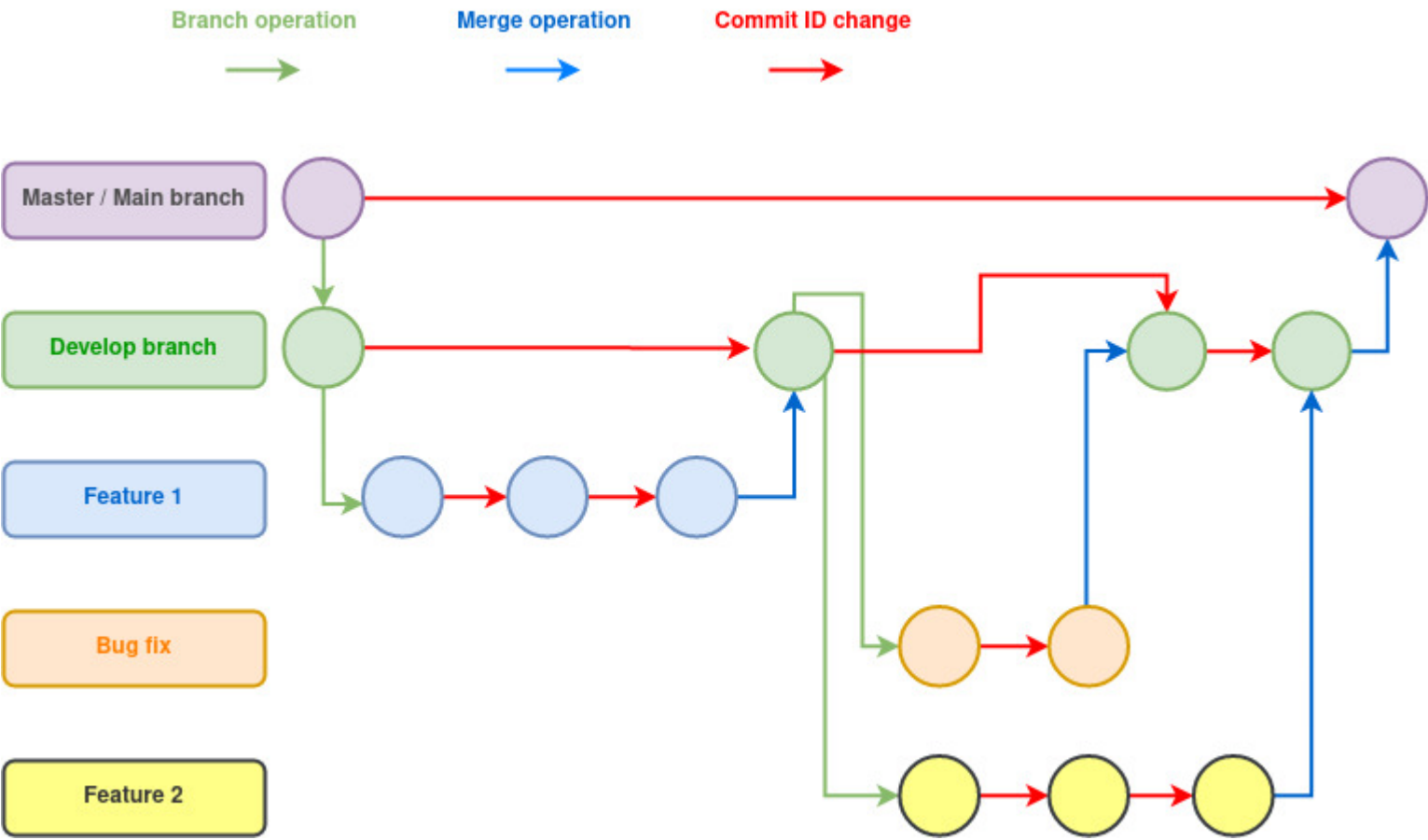


- Go to the directory where you want to clone the project. Open the terminal. Make sure you are in the same directory from the terminal. Then write the following command:

```
$ git clone -b "your_branch_name" paste_the_URL_here
```

- When it's complete, You will find a new folder named after the project name.

Branching & Merging Workflow



• Branching

- Developers will not commit directly to the `main / develop` branch. Instead of committing directly to the `main / develop` branch developers will create a new branch under the `develop` branch and commit there.
- During a sprint, development works will be merged into the `develop` branch
- At the end of a sprint, the `develop` branch will be merged into the `main` branch, **ONLY** if there are no problems with the `develop` branch.

• Creating a new branch

- To create a new branch go to the project folder. Make sure you are on the `develop` branch. To do that open the terminal and write the following command:

```
$ git branch
```

- In case you find yourself in another branch, just switch to `develop` branch using the following command:

```
$ git checkout develop
```

- Now, to create a branch under the `develop` branch use the following command:

```
$ git checkout -b name/reason/branch_name develop
# the above command creates a new branch and automatically switches to the new branch.
```

◦ Branch naming strategy

- `developer-name` = as for marjan
- `reason` = `issue/feature/docs/design/bugfix`
- `branch_name` = feel free to give a relevant name to your branch.

```
developer-name/reason/branch-name
```

• Merging

- To merge your codes, you need to create a merge request.
- **Checklist before creating a merge request**
 - Analyze your codes with Code Quality Testing Tools.
 - Run unit tests.

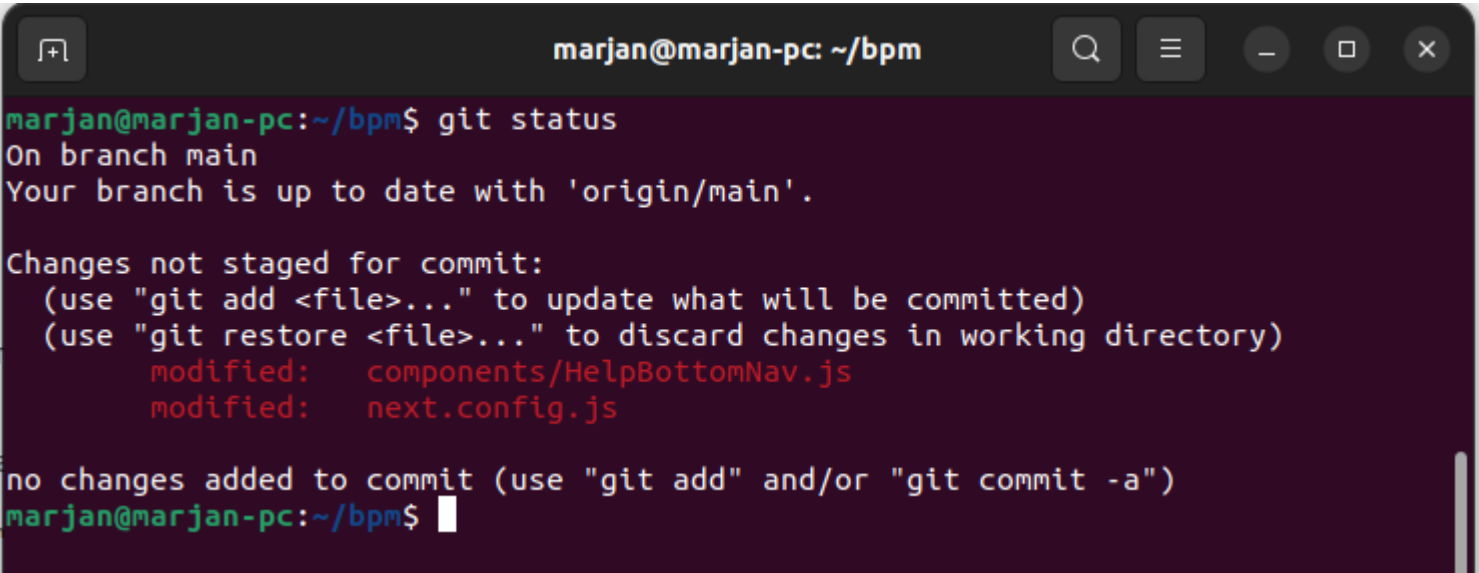
- When writing markdown files, limit each line to 120 characters.
- Developers must do a spell check before creating the MR.

◦ **Git add, commit & push steps**

- Go to the project folder. Open the terminal and use the following commands:

```
$ git status
```

example:



- Using the above command you will be able to find the modified/updated files. Then just simply use the following command to make the update/change inclusive in the next commit.

```
$ git add give_file_path_here
```

- Now, to save all the changes, along with a brief description about what changes have been made, use the following command:

```
$ git commit -m 'add_commit_message_here'
# you can add and commit your code as many times as after you like
```

- **Follow the Git Commit Guideline**

- Commit **often**
- Each commit should contain a small code change
- A typical feature/bug-fix implementation might include many commits
- Commit message guidelines:
 - [TheServerSide: How to write a Git commit message properly with examples](#)
 - [Chris Beams: How to Write a Git Commit Message](#)

- To push your codes the first time, use the following command:

```
$ git push -u origin branch_name
```

- From next time when you need to push codes, just use `git push` :

```
$ git push
# don't need to use $ git push -u origin branch_name again.
```

• **Creating a new Merge request**

- After git push, you should find something like the following:

```
marjan@marjan-pc: ~/Documents/bpm
marjan@marjan-pc:~/Documents/bpm$ git push -u origin marjan/bugfix/voice-support-for-firefox
Username for 'http://medusa.ghit.com': marjan.itsd@goldenharvestbd.com
Password for 'http://marjan.itsd@goldenharvestbd.com@medusa.ghit.com':
Branch 'marjan/bugfix/voice-support-for-firefox' set up to track remote branch 'marjan/bugfix/voice-support-for-firefox' from 'origin'.
Everything up-to-date
marjan@marjan-pc:~/Documents/bpm$
```

- If you can't find that, no worries simply visit <https://pms.ghitbd.net/> and go to your project. You should find the alert box "You pushed to ... at ... minutes ago":

BPM > bpm-web

✔

You pushed to [marjan/bugfix/voice-support-for-firefox](#) at [BPM / bpm-web](#) 34 minutes ago

Create merge request

B

bpm-web

Project ID: 51

Leave project

3 Commits

2 Branches

0 Tags

254.8 MB Files

255.2 MB Storage

- Click on the button to create a merge request.
- Feel free to give a relevant title to your MR. For example:
 - Added yaml file for ci pipeline
 - Removed maven wrapper files
 - Added test coverage, etc
- Add a clear description of the MR explaining its objective and how that objective is achieved.
- Under the Title field, you will find "Start the title with Draft:". **Click it!** This will create a Draft merge request.

BPM > bpm-web > Merge requests > New

New merge request

From `marjan/bugfix/voice-support-for-firefox` into `main` [Change branches](#)

Title *

Voice support for firefox

Start the title with **Draft:** to prevent a merge request draft from merging before
Add [description templates](#) to help your contributors to communicate effectively!

- You should find the Assignee and Reviewer field below. Select them accordingly and click on the Create merge request button. Finally, your MR is ready for review.
- **Checklist after creating a merge request**
 - If the Git pipeline failed, MR will not be reviewed.
 - Inform the reviewer that the merge request (MR) is ready for review.
 - **Responding to MR response:**
 - Unsolved issues can be tracked using this button:

https://pms.ghitbd.net/bpm/bpm-docs/-/wikis/Git-&-Gitlab

7/10

BPM > bpm-web > Merge requests > 11

Open Created 1 minute ago by  **Abdur Rab Marjan** Maintainer

Edit

Mark as ready



Draft:Voice support for firefox

Overview 0 Commits 4 Changes 130+**Request to merge** [marjan/bugfix/voice...](#) into **main**


The source branch is 3 commits behind the target branch

Open in Web IDE

Check out branch



Approve

Approval is optional 

Merge

Merge blocked: merge conflicts must be resolved.

Resolve locally



0



0



Oldest first ▾

Show all activity ▾

**Abdur Rab Marjan** @marjan added **Working** label 1 minute ago**Abdur Rab Marjan** @marjan requested review from [@marjan](#) 1 minute ago**Abdur Rab Marjan** @marjan assigned to [@suhel.itsd](#) 1 minute ago

- Checkout, merge and review locally:

Check out, review, and merge locally

Step 1. Fetch and check out this merge request's feature branch:

```
git fetch origin
git checkout -b 'marjan/bugfix/voice-support-for-firefox' 'origin/marjan/bugfix/voice-su'
```

Copy commands

Step 2. Review the changes locally.

Step 3. Merge the feature branch into the target branch and fix any conflicts. [How do I fix them?](#)

```
git fetch origin
git checkout 'main'
git merge --no-ff 'marjan/bugfix/voice-support-for-firefox'
```

Step 4. Push the target branch up to GitLab.

```
git push origin 'main'
```

Tip: You can also check out merge requests locally. [Learn more.](#)

- **Replying to MR Comments:**
 - **Don't agree to everything.**
 - Be polite.
 - Give a reason and an example.
 - If you agree, do the change, push it, and post a reply like
 - "Done", if the code change is small/obvious.
 - Otherwise, describe what you have done to resolve the comment. The description should help the reviewer to understand what code changes were done.
 - If you disagree, explain why on the MR page. It's always better to document it in words compared to discussing it over WhatsApp/Slack.
 - **Every MR comment must have a response.**
 - **Do not mark discussions as Resolved. The reviewer will do that, once they are happy with the fix.**

- **Reviewing a Merge request**

[Code Review Developer Guide](#)

Git commands for developers

```
# To know which branch you are in.
$ git branch

# To pull to get the latest commits from the remote repository
$ git pull

# To create a new branch, use the command below-
$ git checkout -b branch-name

#To switch from one branch to another, use the command below-
$ git checkout branch-name

# push your code
$ git push

# pull in the new changes (code from other developers) on the develop branch
# to your local machine. This approach reduces the chance of merge conflicts
$ git fetch origin develop: develop
# this command will pull in the new changes on the remote develop branch to your machine

# merge the new code into your local branch.
$ git merge develop

# OR if you have a lot of whitespaces (formatting) changes you can abort the previous merge
$ git merge --abort
# and then try
$ git merge -Xignore-space-change develop

# git pull from another branch to the current branch
$ git pull origin branch-name

# Deleting a remote branch
$ git push origin -d <branch>

# Deleting a local branch
$ git branch -d <branch>

# Record the current state of the working directory
$ git stash

## Clone from one repo and push to other repo

# Clone from gitlab bpm web repo
$ git clone https://gitlab.com/bpm4/bpm.git

# Create branch for bug fix
$ git checkout -b marjan/bugfix/voice-support-for-firefox

# Stage and commit all file
$ git add .
$ git commit -m "Bugfix git test"

# Set new repo url for push to pms bpm web repo
$ git remote set-url origin https://pms.ghitbd.net/bpm/bpm_frontend.git

# Before push first pull or fetch (It will reduce merge conflict issue)
$ git pull
$ git push

# Push new medusa repo with already created branch
$ git push --set-upstream origin marjan/bugfix/voice-support-for-firefox
```

Resources

- Git-cheat-sheet: <https://about.gitlab.com/images/press/git-cheat-sheet.pdf>