



Github Session 08

Summary 05-03-2022

- The workspace, also known as the working directory, is where you modify, create, and delete files for your project. It's the directory on your local machine where you have all your project files.
- The Staging area is the middle ground between what you have done to your files (also known as the working directory) and what you had last committed (the HEAD commit). The staging maintains the index for what changes has been made in the files.
- The commit area is where your changes are permanently stored in the Git repository.
- Whenever we last commit, the information about it is stored in a variable known as the HEAD variable,
- If you are doing any operations in the commit area, that operation will always check where your HEAD is.
- GitViz is a tool that provides a graphical representation of branches, commits, and their relationships, making it easier to understand the branching structure and the evolution of a project over time.
- To download the zip file of gitviz you can visit to the link <https://github.com/Readify/gitViz/releases> .
- To understand the GitViz, lets create a new file and commit it to the github.

```
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  a.txt

nothing added to commit but untracked files present (use "git add" to track)

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop/ws11 (master)
$ git add a.txt

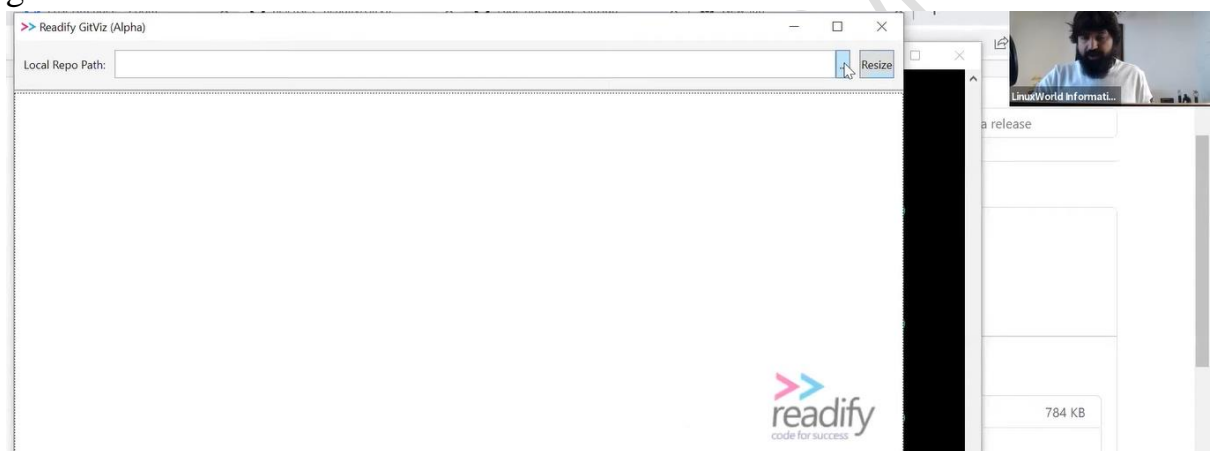
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop/ws11 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   a.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop/ws11 (master)
$ git commit a.txt -m "commit #1"
```

- Add the file location in the GitViz to visualize what we have done in the git.



- Lets create one more file b.txt, as soon as we commit this file, we can see some live changes in the GitViz.

Local Repo Path: C:\Users\Vimal Daga\Documents\gitws-workshop\ws11

HEAD -> master

8435108

871fede

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop/ws11 (master)
$ git log --oneline
871fede (HEAD -> master) commit #1

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop/ws11 (master)
$ touch b.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop/ws11 (master)
$ git add b.txt

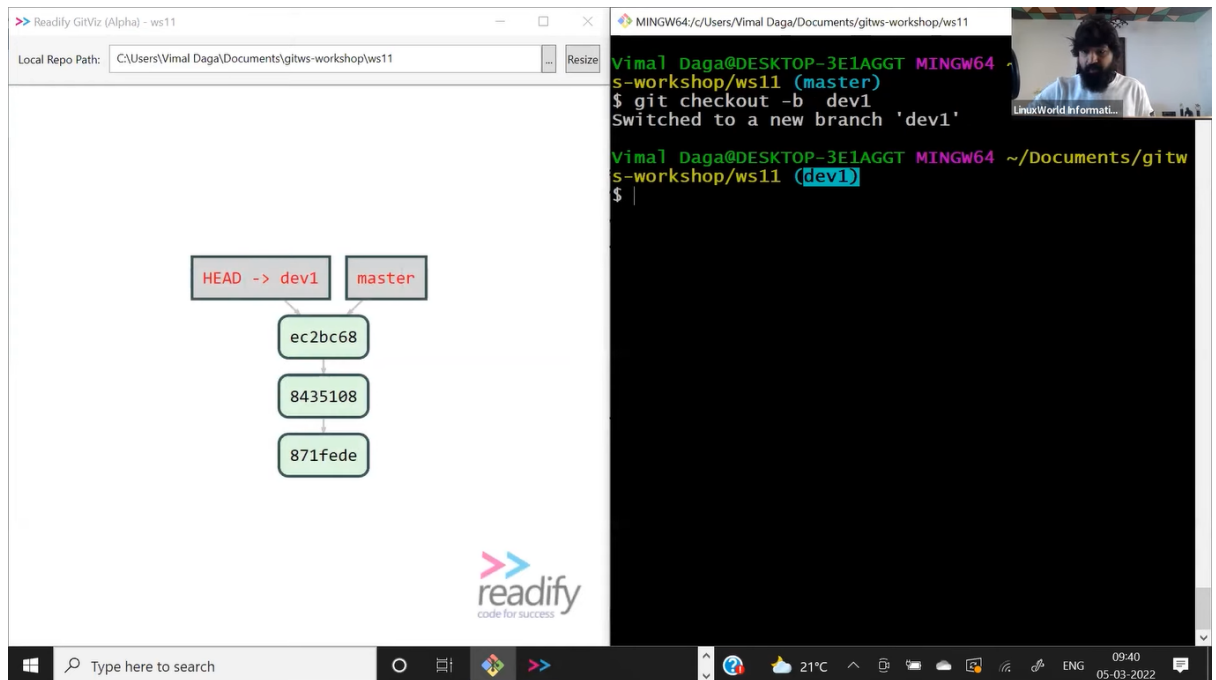
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop/ws11 (master)
$ git commit b.txt -m "commit #2"
[master 8435108] commit #2
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 b.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitws-workshop/ws11 (master)
$
```

➤ We can see that a new branch is added now.

- Now if we create a new branch(dev1), the entire timeline or the master branch in the commit area will come to this new branch also.

[Mastering Git and Github]

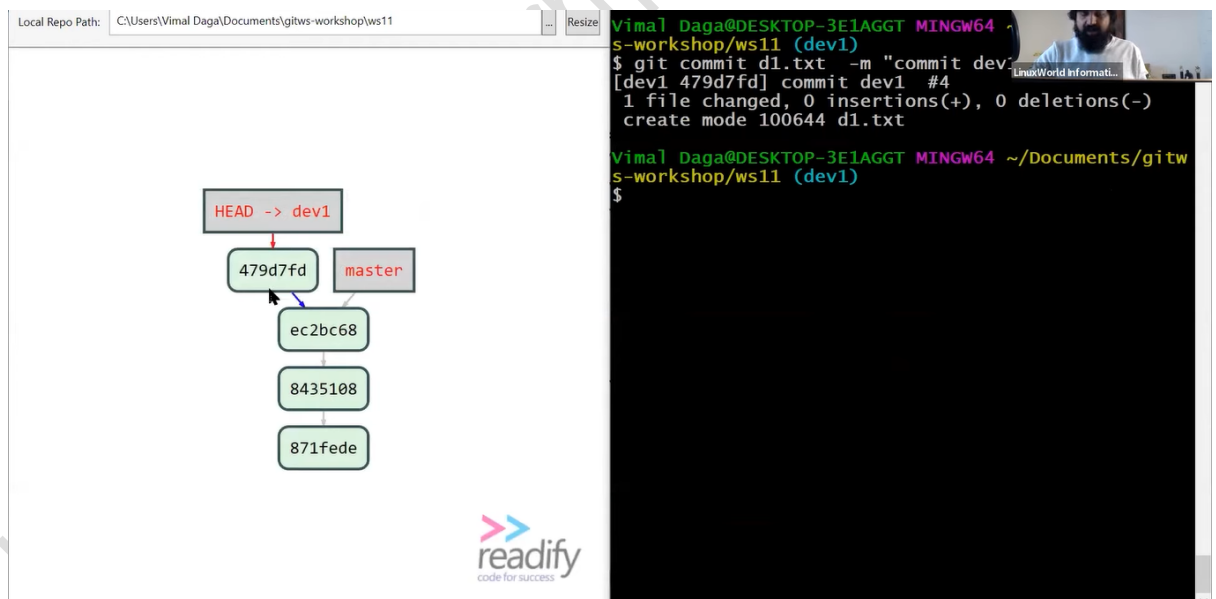


The screenshot shows the Readify GitViz interface on the left and a terminal window on the right. The GitViz interface displays the local repository path as `C:\Users\Vimal Daga\Documents\gitws-workshop\ws11`. The commit history is shown as a vertical stack of three commits: `ec2bc68`, `8435108`, and `871fede`. The `HEAD` pointer is shown pointing to the `dev1` branch, and the `master` branch is also shown. The terminal window shows the following commands and output:

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~\Documents\gitw
s-workshop\ws11 (master)
$ git checkout -b dev1
Switched to a new branch 'dev1'

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~\Documents\gitw
s-workshop\ws11 (dev1)
$
```

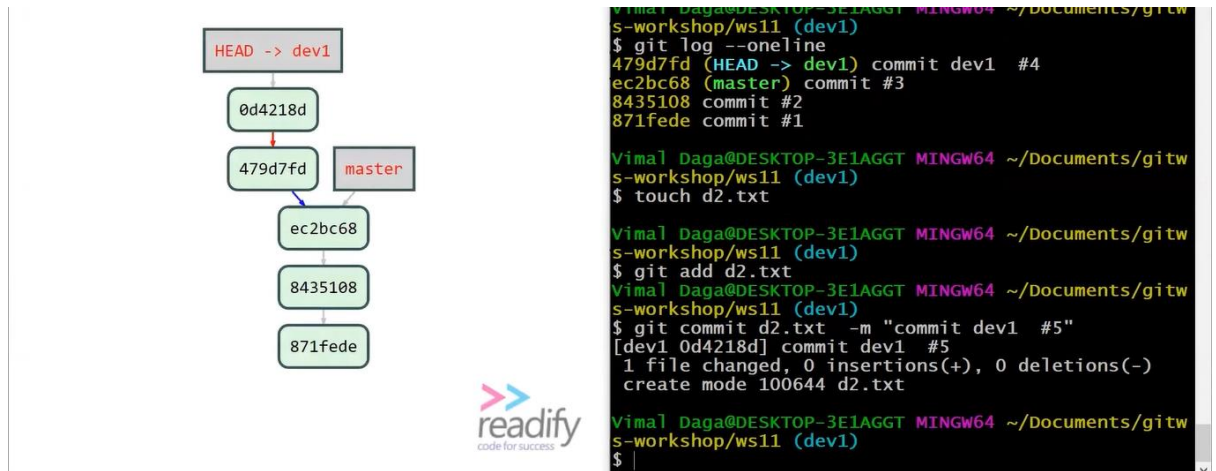
- Now if we commit a new file from the new branch, what we will see is that the master branch will still point to the previous commit and the new branch(`dev1`) is one commit ahead from the master.



The screenshot shows the Readify GitViz interface on the left and a terminal window on the right. The GitViz interface displays the local repository path as `C:\Users\Vimal Daga\Documents\gitws-workshop\ws11`. The commit history is shown as a vertical stack of four commits: `479d7fd`, `ec2bc68`, `8435108`, and `871fede`. The `HEAD` pointer is shown pointing to the `dev1` branch, and the `master` branch is also shown. The terminal window shows the following commands and output:

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~\Documents\gitw
s-workshop\ws11 (dev1)
$ git commit d1.txt -m "commit dev1"
[dev1 479d7fd] commit dev1 #4
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 d1.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~\Documents\gitw
s-workshop\ws11 (dev1)
$
```



The diagram on the left shows a vertical sequence of commit hashes: 0d4218d, 479d7fd, ec2bc68, 8435108, and 871fede. A box labeled 'HEAD -> dev1' points to 0d4218d. A box labeled 'master' points to 479d7fd. The terminal on the right shows the following commands and output:

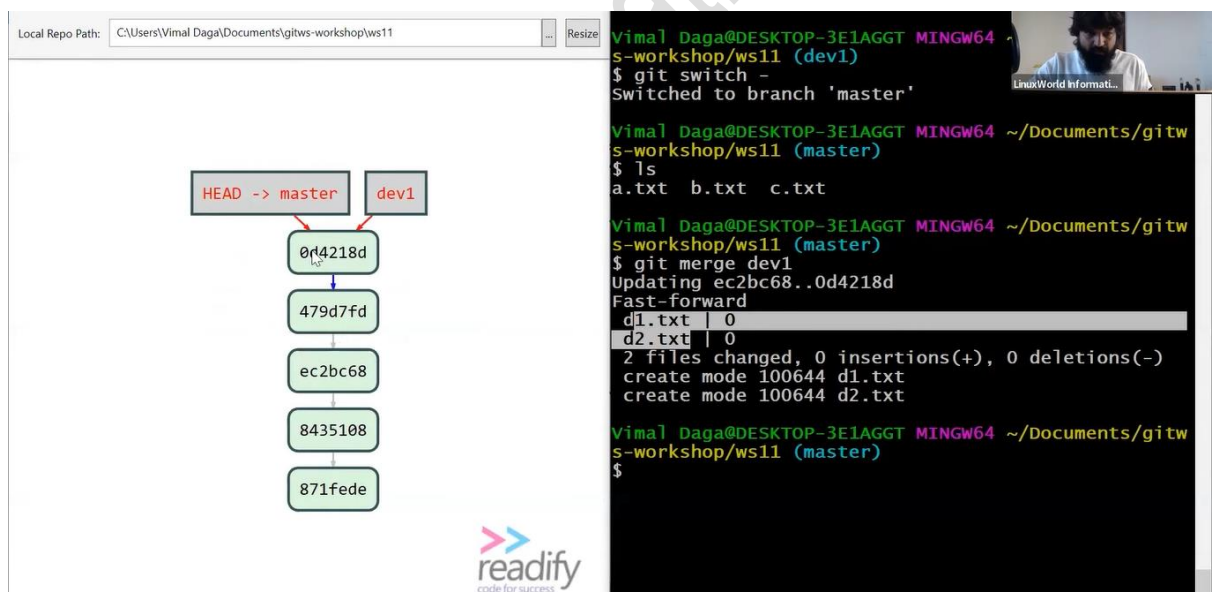
```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (dev1)
$ git log --oneline
479d7fd (HEAD -> dev1) commit dev1 #4
ec2bc68 (master) commit #3
8435108 commit #2
871fede commit #1

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (dev1)
$ touch d2.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (dev1)
$ git add d2.txt
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (dev1)
$ git commit d2.txt -m "commit dev1 #5"
[dev1 0d4218d] commit dev1 #5
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 d2.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (dev1)
$
```

- The ***git switch*** – command is used to switch the branch with the previous branch.
- To merge the new branch and the master branch we use the command ***git merge <new branch name>***



The diagram on the left shows a vertical sequence of commit hashes: 0d4218d, 479d7fd, ec2bc68, 8435108, and 871fede. A box labeled 'HEAD -> master' points to 0d4218d. A box labeled 'dev1' points to 479d7fd. The terminal on the right shows the following commands and output:

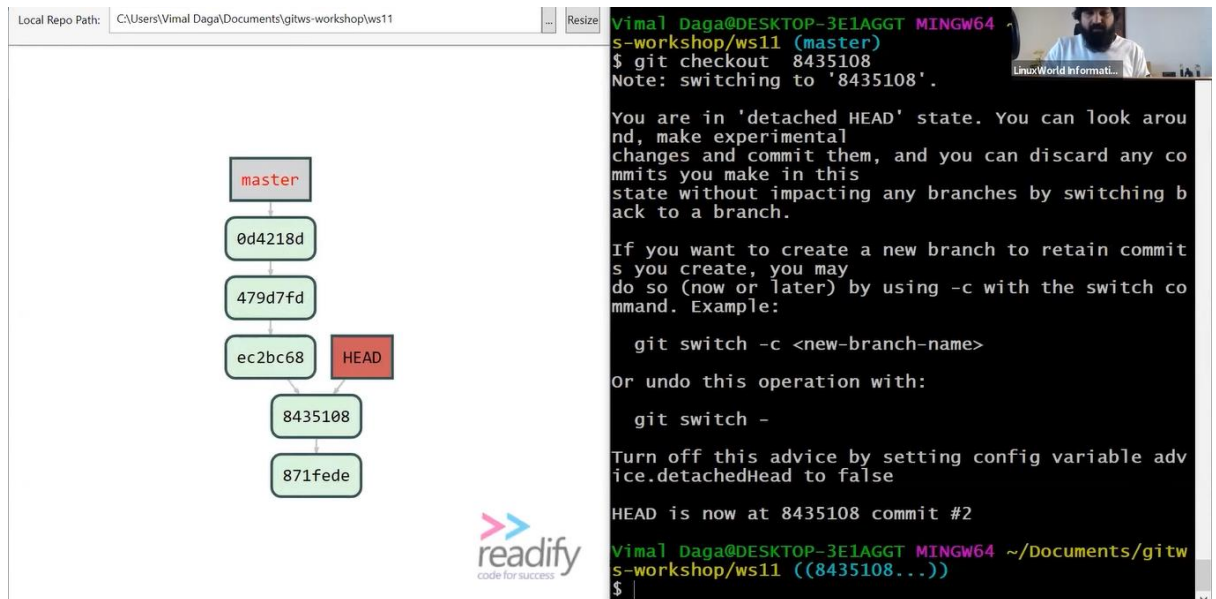
```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
s-workshop/ws11 (dev1)
$ git switch -
Switched to branch 'master'

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (master)
$ ls
a.txt b.txt c.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (master)
$ git merge dev1
Updating ec2bc68..0d4218d
Fast-forward
 d1.txt | 0
 d2.txt | 0
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 d1.txt
create mode 100644 d2.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (master)
$
```

- We can change the HEAD pointer position by using ***git checkout <commit id>***



The screenshot shows a Git GUI window with a local repository path of C:\Users\Vimal Daga\Documents\gitws-workshop\ws11. The commit history is displayed as a vertical chain: master (0d4218d) -> 479d7fd -> ec2bc68 -> 8435108 (HEAD) -> 871fede. A terminal window on the right shows the command `git checkout 8435108` being executed, resulting in a 'detached HEAD' state. The terminal also displays instructions on how to create a new branch and how to switch back to a branch.

```
Local Repo Path: C:\Users\Vimal Daga\Documents\gitws-workshop\ws11
master
0d4218d
479d7fd
ec2bc68 HEAD
8435108
871fede

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
s-workshop/ws11 (master)
$ git checkout 8435108
Note: switching to '8435108'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

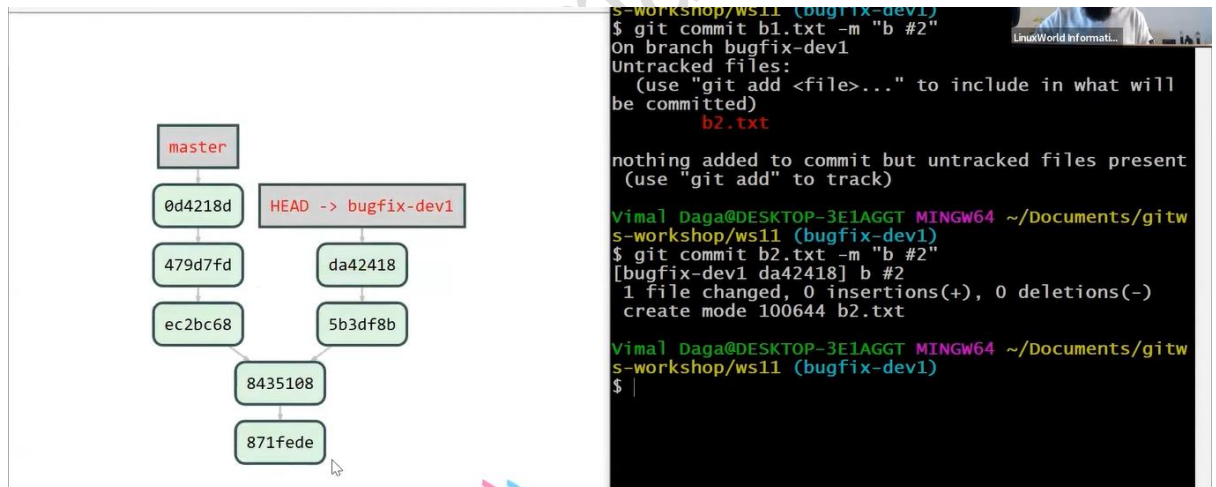
  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 8435108 commit #2
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 ((8435108...))
$
```

➤ We can see that the HEAD is pointing to the 8435108.

- Now if we create a new branch from the new HEAD position and add some new files in it, we can see that a diversion is created in the tree and files will be added in the new branch.



The screenshot shows a Git GUI window with a local repository path of C:\Users\Vimal Daga\Documents\gitws-workshop\ws11. The commit history is displayed as a branching diagram: master (0d4218d) -> 479d7fd -> ec2bc68 -> 8435108 -> 871fede. A new branch, bugfix-dev1, is created from 8435108 and contains commits da42418 and 5b3df8b. The HEAD is now pointing to bugfix-dev1. A terminal window on the right shows the command `git commit b1.txt -m "b #2"` being executed, resulting in a new commit on the bugfix-dev1 branch.

```
Local Repo Path: C:\Users\Vimal Daga\Documents\gitws-workshop\ws11
master
0d4218d
479d7fd
ec2bc68
8435108
871fede
HEAD -> bugfix-dev1
da42418
5b3df8b

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
s-workshop/ws11 (bugfix-dev1)
$ git commit b1.txt -m "b #2"
On branch bugfix-dev1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  b2.txt

nothing added to commit but untracked files present (use "git add" to track)
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (bugfix-dev1)
$ git commit b2.txt -m "b #2"
[bugfix-dev1 da42418] b #2
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 b2.txt
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/gitw
s-workshop/ws11 (bugfix-dev1)
$
```

- If we merge both the branches now, we will see a different kind of Tree with a special merging.

The screenshot displays a Git GUI interface. On the left, a commit graph shows the following structure:

- HEAD -> master (a2d3893)
- 0d4218d (parent of a2d3893)
- 479d7fd (parent of 0d4218d)
- ec2bc68 (parent of 479d7fd)
- 8435108 (parent of 479d7fd)
- 871fede (parent of 8435108)
- bugfix-dev1 (branch, parent of a2d3893)
- da42418 (parent of bugfix-dev1)
- 5b3df8b (parent of da42418)

On the right, a terminal window shows the following commands and output:

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~\Documents\gitw
s-workshop/ws11 (master)
$ ls
a.txt b.txt c.txt d1.txt d2.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~\Documents\gitw
s-workshop/ws11 (master)
$ git merge bugfix-dev1
Merge made by the 'ort' strategy.
 b1.txt | 0
 b2.txt | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 b1.txt
 create mode 100644 b2.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~\Documents\gitw
s-workshop/ws11 (master)
$ ls
a.txt b1.txt c.txt d2.txt
b.txt b2.txt d1.txt

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~\Documents\gitw
s-workshop/ws11 (master)
$
```

- This kind of merging strategy is known as the ort strategy.